

CURSO PRACTICO DEL ORDENADOR PERSONAL,
EL MICRO Y EL MINIORDENADOR

Editorial Delta, S.A.

MI COMPUTER CURSO PRACTICO
DEL ORDENADOR PERSONAL, EL MICRO Y EL MINIORDENADOR

Publicado por Editorial Delta, S.A., Barcelona

Volumen IX-Fascículo 107

Director: José Mas Godayol
Director editorial: Gerardo Romero
Jefe de redacción: Pablo Parra
Coordinación editorial: Jaime Mardones, Francisco Martín
Asesor técnico: Ramón Cervelló

Redactores y colaboradores: G. Jefferson, R. Ford, F. Martín, S. Tarditti, A. Cuevas, F. Blasco
Para la edición inglesa: R. Pawson (editor), D. Tebbutt (consultant editor), C. Cooper (executive editor), D. Whelan (art editor), Bunch Partworks Ltd. (proyecto y realización)

Realización gráfica: Luis F. Balaguer

Redacción y administración:
Aribau, 185, 1.°, 08021 Barcelona
Tel. (93) 209 80 22 - Télex: 93392 EPPA

MI COMPUTER, *Curso práctico del ordenador personal, el micro y el miniordenador*, se publica en forma de 120 fascículos de aparición semanal, encuadernables en diez volúmenes. Cada fascículo consta de 20 páginas interiores y sus correspondientes cubiertas. Con el fascículo que completa cada uno de los volúmenes, se ponen a la venta las tapas para su encuadernación.
El editor se reserva el derecho de modificar el precio de venta del fascículo en el transcurso de la obra, si las circunstancias del mercado así lo exigieran.

ISBN: 84-85822-83-8 (fascículo) 84-7598-181-X (tomo 9)
84-85822-82-X (obra completa)
Depósito Legal: B. 52-84

Fotocomposición: Tecfa, S.A., Pedro IV, 160, Barcelona-5
Impresión: Cayfosa, Santa Perpètua de Mogoda (Barcelona) 048602
Impreso en España-Printed in Spain-Febrero 1986

Editorial Delta, S.A., garantiza la publicación de todos los fascículos que componen esta obra.

Distribuye para España: Marco Ibérica, Distribución de Ediciones, S.A., Carretera de Irún, km 13,350. Variante de Fuencarral, 28034 Madrid.
Distribuye para Colombia: Distribuidoras Unidas, Ltda., Transversal 93, n.° 52-03, Bogotá D.E.
Distribuye para México: Distribuidora Intermex, S.A., Lucio blanco, n.° 435, Col. San Juan Tlihuaca, Azcapotzalco, 02400, México D.F.
Distribuye para Venezuela: Distribuidora Continental, S.A., Edificio Bloque Dearmas, final Avda. San Martín con final Avda. La Paz, Caracas 1010.

Pida a su proveedor habitual que le reserve un ejemplar de MI COMPUTER. Comprando su fascículo todas las semanas y en el mismo quiosco o librería, Vd. conseguirá un servicio más rápido, pues nos permite realizar la distribución a los puntos de venta con la mayor precisión.

Servicio de suscripciones y atrasados (sólo para España)

Las condiciones de suscripción a la obra completa (120 fascículos más las tapas, guardas y transferibles para la confección de los 10 volúmenes) son las siguientes:

a) Un pago único anticipado de 27 105 ptas. o bien 10 pagos trimestrales anticipados y consecutivos de 2 711 ptas. (sin gastos de envío).

b) Los pagos pueden hacerse efectivos mediante ingreso en la cuenta 6.850.277 de la Caja Postal de Ahorros y remitiendo a continuación el resguardo o su fotocopia a Editorial Delta, S. A. (Aribau, 185, 1.°, 08021 Barcelona), o también con talón bancario remitido a la misma dirección.

c) Se realizará un envío cada 12 semanas, compuesto de 12 fascículos y las tapas para encuadernarlos.

Los fascículos atrasados pueden adquirirse en el quiosco o librería habitual. También pueden recibirse por correo, con incremento del coste de envío, haciendo llegar su importe a Editorial Delta, S.A., en la forma establecida en el apartado b).

No se efectúan envíos contra reembolso.





Imagen por números

El Print-Technik pone en nuestras manos la más avanzada tecnología en el campo de la digitalización de la imagen

Marcus Wilson-Smith

Un digitalizador de video toma una señal proveniente de cualquier fuente de video (por lo general, de una cámara, pero también de un disco láser o una grabadora de video) y convierte las tensiones analógicas de la señal en un gran conjunto de números que representan la imagen. Estos números se pueden almacenar en la memoria de un ordenador, visualizar en gráficos de alta resolución, tratar mediante software, volcar a una impresora, etc.

Los digitalizadores de video pueden tener múltiples usos, desde aplicaciones serias, como sistemas de alarmas, a ideas divertidas como pueden ser "imágenes por ordenador" y distintivos de personas. Print-Technik cita el ejemplo de un peluquero que digitaliza imágenes de sus clientes. Utilizando un lápiz óptico y un paquete para gráficos, puede mostrarle al cliente varios estilos de peinado para ayudarles a escoger el adecuado.

La unidad de Print-Technik, sin embargo, no parece tener capacidad para realizar todo esto. Se trata de una pequeña caja que se introduce en la puerta para el usuario del Commodore 64. Un conector de video estándar permite la conexión con su fuente de video; no hay ningún otro cable, ya que la unidad es alimentada por el propio ordenador. Los únicos otros "controles" son tres pequeños tornillos regulables que varían el brillo, el contraste y la anchura de la imagen digitalizada.

La unidad se controla mediante un paquete de software que se suministra en disco. Éste es un programa funcional y bien construido, pero hace muy poco en cuanto a explotar las auténticas posibilidades de la unidad. El programa presenta un menú amable del cual se seleccionan las opciones desplazando un puntero en forma de mano mediante las teclas del cursor y pulsando RETURN. Se puede optar por digitalizar la señal, ver la imagen que esté actualmente en la memoria, guardarla en disco o imprimirla. Se admiten distintas impresoras, incluyendo las unidades 801 y 1525 de Commodore. Se puede cargar un programa especial para impresoras llamado "16 colores" para imprimir imágenes en hasta 16 colores utilizando una de las cinco principales impresoras matriciales en color. Las imágenes en pantalla se limitan a cuatro colores, pero el digitalizador puede resolver hasta 256 tonalidades.

Como cabría esperar, el software está diseñado para ser empleado conjuntamente con otros programas. Una opción LIGHTPEN llama al propio software para gráficos con lápiz óptico de Print-Technik, si bien el mismo no se suministra como estándar. Lo que es más importante, la imagen de la memoria se puede guardar en disco en un formato apto para dos conocidos paquetes artísticos para el Commodore 64: *Koala-pad* y *Paintmagic*. Los artistas y diseñadores que utilicen estos programas encontrarán que el digitalizador de video representa una valiosa adición a su gama de herramientas. Todas las rutinas del software para guardar y cargar imágenes usan cargadores rápidos para mejorar el rendimiento de la unidad de disco 1541.

En la actualidad, la digitalización de una imagen emplea cuatro segundos, lo que limita las aplicaciones potenciales para la unidad y la hace más difícil de utilizar. El sistema de menor precio que puede formarse se basaría en una cámara de televisión de circuito cerrado similar a las que se utilizan para vigilancia. Estas cámaras no poseen pantalla/visor interno. Por consiguiente, es casi preceptivo tener disponible una pantalla de video normal de modo que el operador pueda centrar el sujeto y enfocar la cámara visualmente antes de desconectarla de la pantalla y conectarla al digitalizador, lista ya para captar la imagen. De modo que, si bien el sistema funciona sin monitor externo, se convierte en una cuestión de ensayo y error, reduciendo en gran medida la calidad de los resultados.

Los cuatro segundos necesarios para producir una imagen también hacen difícil captar una imagen en movimiento. Para obtener imágenes de per-

Técnicas visuales
El digitalizador de video Print-Technik para el Commodore 64 capta imágenes de una cámara o grabadora de video y las digitaliza, permitiendo visualizarlas en la pantalla en cuatro tonalidades, almacenarlas en disco o imprimirlas. Una facilidad particularmente interesante permite guardar las imágenes digitalizadas en disco en formatos que utilizan otros paquetes para gráficos, como el *Koala-pad*. El paquete incluye un pequeño cartucho que se enchufa en la puerta para el usuario y varios paquetes de calidad en disco para soportar el hardware

DIGITALIZADOR DE VIDEO PRINT-TECHNIK

DIMENSIONES

80 × 65 × 20 mm

CONTROLES EXTERNOS

Brillo, contraste y anchura de imagen

RESOLUCION

256 × 256 pixels en 256 tonalidades. Normalmente las imágenes se presentan en formato de 160 × 200 pixels utilizando cuatro tonalidades

VELOCIDAD

Cuatro segundos por imagen

SOFTWARE

Mediante el software estándar las imágenes se pueden digitalizar, guardar en formato *Koala-pad* o *Printmagic* de 256 × 256, imprimir y volver a colorear. Además, los programas de demostración proporcionan un sistema de alarma y una exhibición de diapositivas, y permiten utilizar imágenes digitalizadas en los propios programas en BASIC

DOCUMENTACION

Un folleto de cuatro páginas no logra cubrir los aspectos más técnicos de la unidad. No obstante, el funcionamiento básico es lo suficientemente sencillo como para que esto no constituya un problema

VENTAJAS

La unidad es compacta, su precio es asequible y trabaja muy bien. El software que se proporciona es adecuado para la mayoría de las aplicaciones simples

DESVENTAJAS

La lenta respuesta significa que la unidad se debe utilizar con una pantalla externa. Los límites de la pantalla de gráficos del Commodore 64 (cuatro colores a 160 × 200 pixels) significan que el paquete estándar no explota cabalmente las capacidades

sonas de calidad, el sujeto ha de permanecer quieto durante los cuatro segundos completos, aunque se pueden crear algunos curiosos efectos mediante el movimiento cuando el digitalizador está en funcionamiento. Hay dos soluciones para este problema. La ideal es utilizar un *captador de cuadro* de video, que congelará la acción electrónicamente. La alternativa es ¡tomar una fotografía y digitalizarla!

Con la fuente correctamente regulada, los únicos otros ajustes que se han de hacer a la imagen captada se efectúan mediante los tres pequeños tornillos de la unidad digitalizadora. Éstos se entregan regulados en niveles óptimos, pero pueden alterarse para satisfacer condiciones extrañas o inusuales. El brillo y el contraste se explican por sí mismos, pero han de regularse mediante ensayo y error, dado que no existe forma inmediata de ver los resultados. El tercer tornillo varía el ancho de la imagen, comprimiendo o ensanchando la imagen resultante. Se puede utilizar para dar las proporciones correctas a una imagen en la pantalla de gráficos del ordenador o bien para producir efectos especiales. Un rostro delgado, por ejemplo, se puede transformar mágicamente con sólo dar vuelta al tornillo.

La imagen digitalizada es de 256 × 256 pixels y, por tanto, en la pantalla de gráficos en color de 160 × 200 del Commodore 64 sólo se puede ver parcialmente. El software de Print-Technik permite explorar la imagen utilizando las teclas del cursor. La imagen se presenta normalmente en cuatro tonos (blanco, gris claro, gris oscuro y negro), aunque el digitalizador trabaja con 256 tonalidades. Puede seleccionarse cualquiera de cuatro colores con las teclas de función empleando el programa suministrado.

También se proporcionan numerosos programas alternativos de muestra, de los cuales el más sofisticado es *Alarm*. Éste permite utilizar la cámara de video conjuntamente con el digitalizador a modo de sereno electrónico. El ordenador digitaliza continuamente la señal, tomando una instantánea de lo que esté enfocando la cámara aproximadamente cada cinco segundos.

Cuando la nueva imagen está lista, se compara con la imagen captada cinco segundos antes y, si es diferente en más de un número preestablecido de lugares, el Commodore 64 hará sonar una alarma. Este programa funcionó muy bien cuando usamos el sistema para vigilar una taza de café. Cuando la cantidad de diferencias que hacen que se dispare la alarma es baja (alrededor de 200), la alarma suena aun cuando alguien se limite a proyectar una ligera sombra sobre la zona vigilada. Obviamente, si estuviera vigilando algunas joyas de valor, el límite de diferencias se habría de establecer más alto, para que la alarma no se disparara por la mera sombra del dueño o cuando disminuyera la luz diurna.

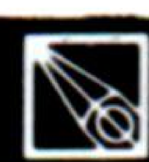
Este programa da una impresión muy favorable de la sensibilidad de la unidad de Print-Technik. Con un Commodore 64 conectado a un verdadero sistema de alarma, habría una posibilidad entre cinco que de un ladrón no fuera detectado, ¡siempre que la escena se pudiera restablecer en el intervalo de un segundo!

Print-Technik también incluye un sencillo programa de proyección de diapositivas que permite exhibir en sucesión en la pantalla imágenes guardadas en formato *Koala-pad*. Por último, hay una rutina que permite presentar imágenes en formato *Koala-pad* desde los propios programas en BASIC usando una llamada SYS.

El digitalizador de Print-Technik transforma al Commodore 64 en una poderosa herramienta para producir imágenes gráficas de gran calidad. El hardware es compacto, produce resultados sorprendentemente buenos, y el software suministrado es amplio, aunque no está tan bien terminado como algunos programas. No obstante, cualquier aplicación práctica de la unidad requiere otros equipos: una cámara de video, una pantalla de video y un programa artístico razonable como el *Paintmagic* o el *Koala-pad*. El atractivo de la unidad se ve limitado, en consecuencia, a quienes la destinen a un uso serio como herramienta para producir gráficos interesantes y detallados o para emplear las aplicaciones más interactivas que hemos descrito.



Posibilidades de impresión
Las imágenes digitalizadas se pueden volcar a una gama de impresoras. Las impresoras MPS 801 y 1525 de Commodore producen en papel una imagen monocromática de cuatro tonalidades, pero con una impresora en color se puede imprimir la imagen en hasta 16 colores



MIDI IN

Finalizamos el proceso de adaptación de la interface MIDI a la gama Amstrad CPC

En nuestro proyecto inicial de una MIDI para los micros Commodore 64 y BBC desarrollamos un programa que permitía que un ordenador actuara como un simple grabador en tiempo real. Debido a que este programa se simplificó para demostrar los principios de la MIDI, los datos entrantes eran objeto de un procesamiento mínimo antes de su almacenamiento. Este procesamiento comprobaba los mensajes del sistema, desechando los que se encontraran y calculando los bytes de temporización a insertar entre los bytes MIDI antes de almacenar los datos en la memoria. Puesto que el intervalo mínimo entre dos bytes MIDI consecutivos es de 320 µs, el bucle de procesamiento para cada byte entrante se mantuvo mucho más corto que este período de tiempo crítico, para asegurar que se llevara a cabo todo el procesamiento y que el programa estuviera en condiciones de recibir el siguiente byte antes de que llegara éste.

Un programa más útil habría llevado a cabo un procesamiento mucho más intensivo de los datos entrantes. Si no fuera por el límite de tiempo de 320 µs impuesto sobre el procesamiento, se podría disponer de facilidades tales como grabación y reproducción simultáneas, visualización en pantalla y superposición de comentarios de canales múltiples. Afortunadamente existe un modo, que se implementa comúnmente en sistemas de ordenador, de "escuchar" los bytes entrantes mientras simultáneamente se procesan datos que ya se han recibido.

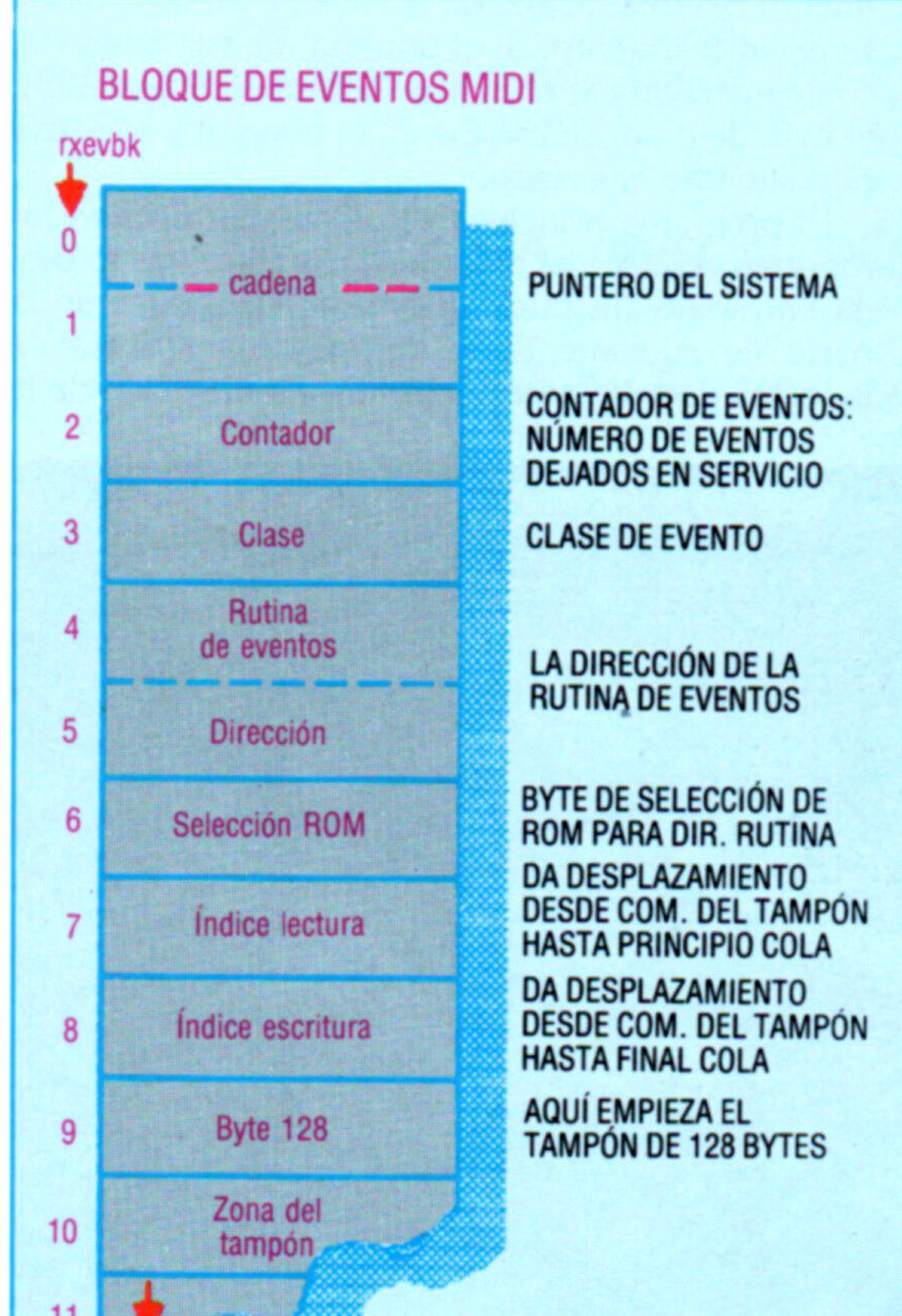
Para demostrar los principios de la escritura de un programa MIDI que posea rutinas de procesamiento largas (de más de 320 µs) veamos un programa para leer datos MIDI entrantes y visualizarlos en la pantalla en tiempo real. Los datos se visualizarán en notación hexadecimal, comenzando cada mensaje MIDI en una nueva línea. Para hacer esto hay que convertir cada byte MIDI recibido en dos dígitos hexadecimales en código ASCII, enviarlos a la pantalla seguidos por dos caracteres de espacio para separarlos del siguiente byte y, si el byte es el último del mensaje MIDI, enviar un retorno de carro. Las cabeceras del sistema operativo, como la exploración del teclado, son importantes en el contexto de una velocidad de transmisión de datos de 320 µs y, por tanto, es probable que una rutina de procesamiento de este tipo emplee más tiempo que el período crítico entre la recepción de cada byte MIDI sucesivo.

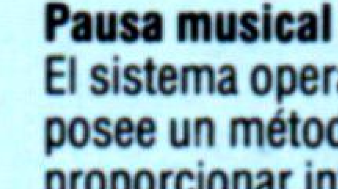
Para no perder datos cuando éstos entran, podríamos usar una interrupción regular en nuestro bucle principal de procesamiento que explorara el registro de estado ACIA y transfiriera todo dato recibido recientemente desde el registro de recepción de datos ACIA a una zona de tamponamiento en la memoria del ordenador. El programa principal podría entonces leer datos de la parte frontal del tampón en un momento adecuado sin que se perdiera ningún dato. Lamentablemente, el intervalo entre interrupciones "de sondeo" necesitaría ser de menos de 320 µs y, como tal, constituye una cabecera de programación inaceptable.

La alternativa a la estructura de interrupciones periódicas es aprovechar la capacidad del ACIA de generar su propia señal de interrupción cada vez que el registro de recepción de datos esté lleno. Por supuesto, necesitamos interceptar la rutina de servicio de interrupciones del Amstrad para distinguir las interrupciones ACIA de otras fuentes de interrupción. Si se detectara una interrupción ACIA, habría que transferir el contenido del registro de recepción de datos.

El programa que vemos aquí utiliza las rutinas de firmware del Amstrad, de las que se ofrece una documentación completa en el manual de firmware y que se puede obtener de Amstrad (SOFT 158). El salto a la rutina para manipular la interrupción ACIA se halla en la posición &003B. La propia rutina de manipulación de interrupción comunica con el programa procesador principal, activando rápidamente un evento, descrito por el bloque de eventos situado en la dirección etiquetada rxevbk.

La zona de tampón está definida como 128 bytes de memoria comenzando desde la dirección rxevbk+9. La rutina comprueba primero si hay algún evento destacable esperando a ser procesado. De no ser así, el tampón está vacío y se inicializan los índices de lectura y escritura. La rutina pasa luego a comprobar si ha habido extralimitación del



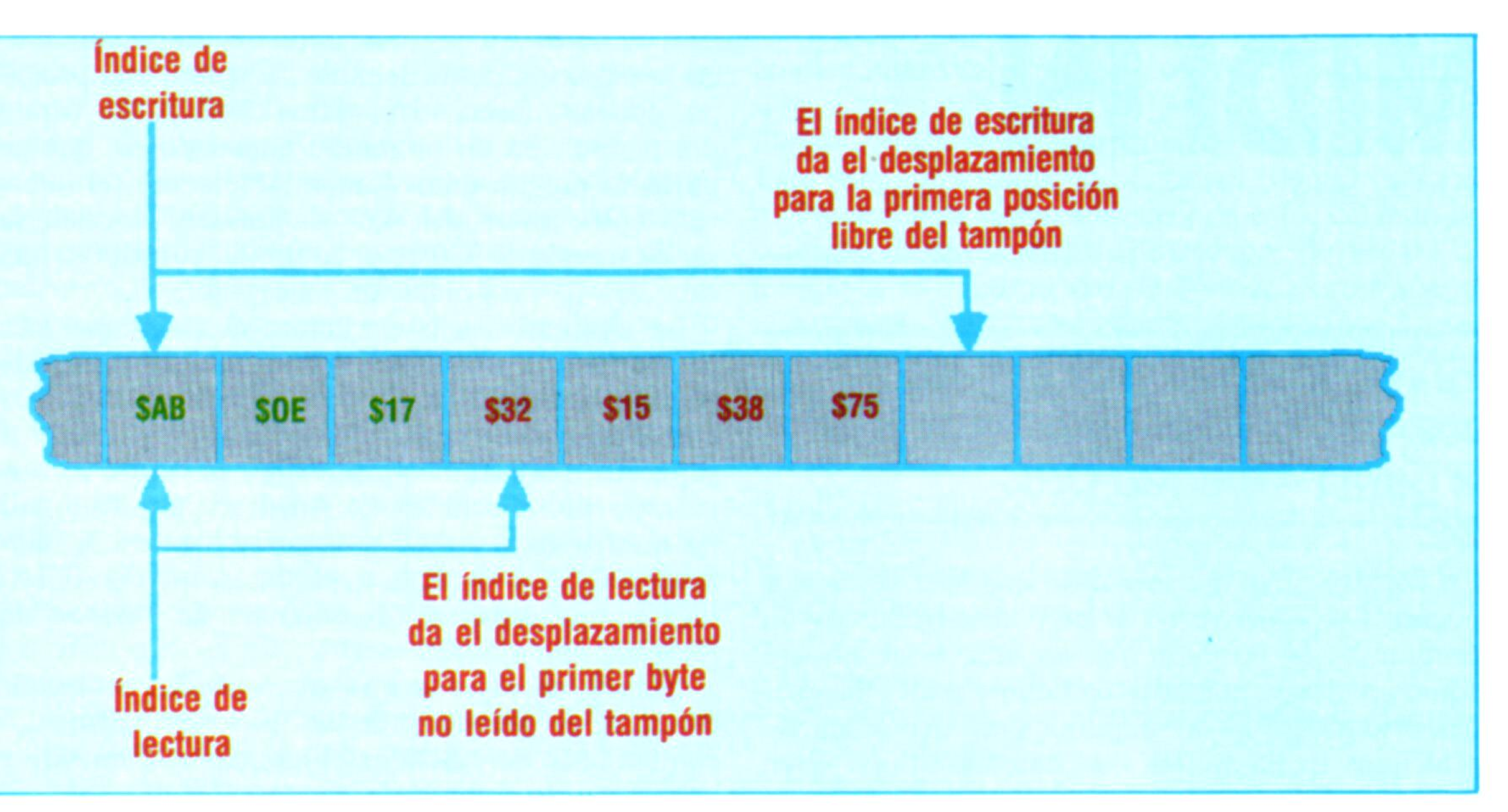


Pausa musical
El sistema operativo Amstrad posee un método de proporcionar interrupciones que se conoce como *manipulación de eventos*. Se puede utilizar un bloque de eventos para manipular interrupciones generadas por el chip ACIA de la interface MIDI cada vez que su registro de recepción de datos esté lleno. Los seis primeros bytes manipulan funciones del sistema, pero los bytes 7 y 8, y los 128 siguientes, se han preparado en nuestro programa para actuar como tampón de entrada de datos

Caroline Clayton

Punteros del tampón

Señalando el camino
Los punteros del principio y el final de la cola del tampón indican los puntos en los cuales se suprimen o insertan datos. Durante la operación, los punteros se irán moviendo progresivamente por la zona de tampón a medida que se vayan suprimiendo datos y añadiendo otros nuevos. Cada vez que el tampón quede vacío (lo que se indicaría cuando los índices READ [lectura] y WRITE [escritura] tuvieran el mismo valor) los índices se restablecen al comienzo del tampón

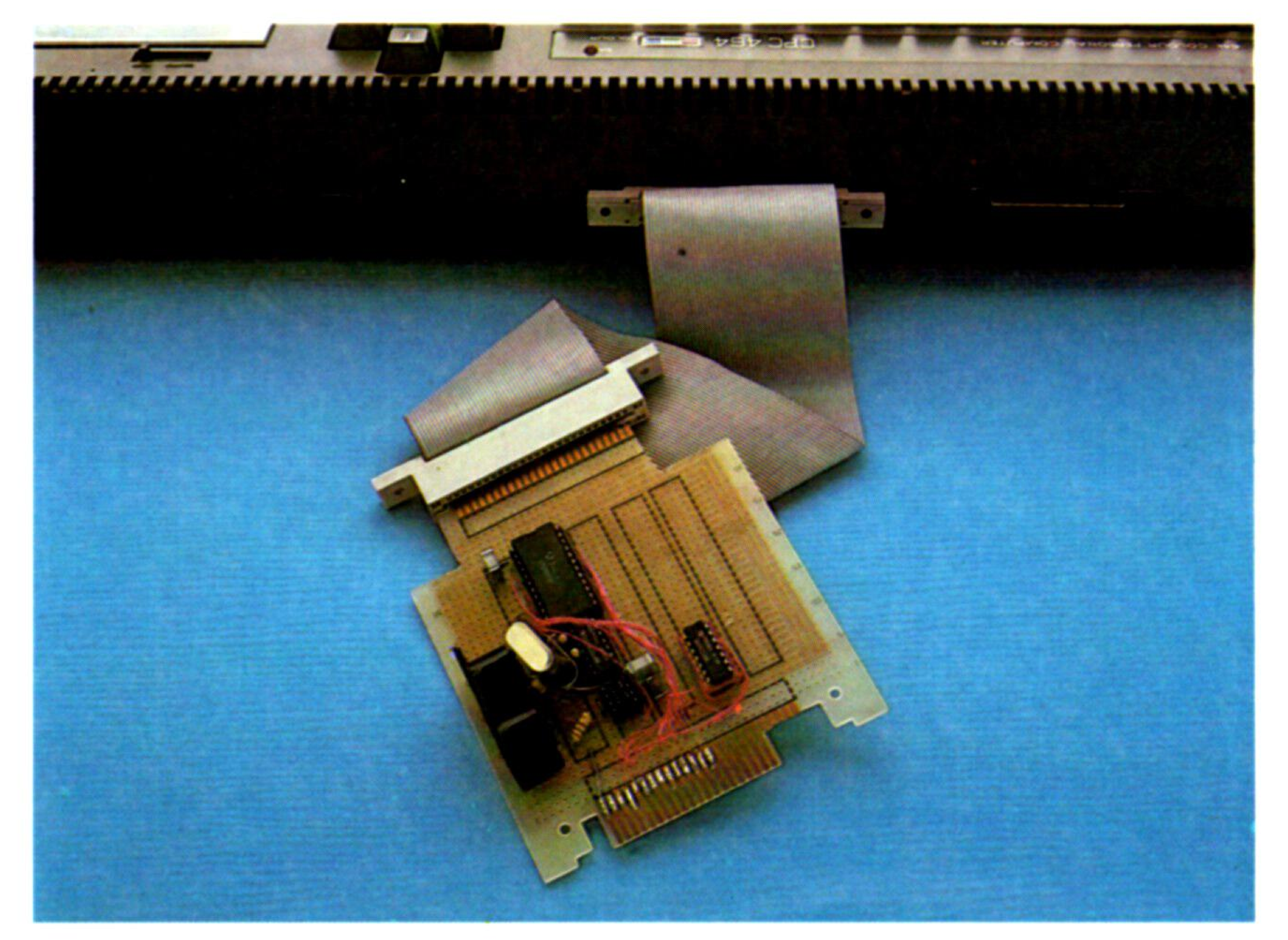


receptor: la llegada de otro byte al ACIA antes de que se haya leído el anterior. La rutina de interrupción debe borrar la fuente de interrupción, ya que de lo contrario se repetiría a sí misma cada vez que se saliera de la rutina manipuladora y se reactivarían las interrupciones, haciendo que la máquina quedara "bloqueada".

La señal de interrupción del ACIA se puede desactivar leyendo el registro de datos ACIA. La rutina calcula luego la dirección de la zona de tampón en la que colocará el byte de datos recibido desde el índice almacenado en el bloque de eventos en rxevbk+8. Este índice es, simplemente, el desplazamiento del final actual de la zona de tampón desde su encabezamiento. Si el tampón no está lleno (tal como sucedería si el índice fuera menor que 128), el byte de datos se escribe en el tampón y se activa rápidamente el evento.

El programa principal simplemente inicializa los registros ACIA y el bloque de eventos, tras lo cual da entrada a un bucle que toma caracteres de la parte de adelante del tampón y los procesa de la forma descrita anteriormente. La dirección de la posición que representa la parte de adelante del tampón se halla a partir del índice de lectura, en la dirección rxevbk+7 dentro del bloque de eventos. Este índice se retiene nuevamente como un desplazamiento desde el comienzo del tampón.

El programa sale si se pulsa cualquier tecla del teclado del ordenador, si el tampón está lleno o si el programa pierde algún byte de datos debido a la extralimitación del receptor.



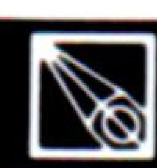
Conectado en interface
Nuestra placa MIDI rediseñada se conecta a la puerta de ampliación del Amstrad a través de un cable plano de 50 vías y conectores de borde. Utilizando zócalos DIN estándares de 5 patillas, se pueden conectar hasta 16 dispositivos de equipo MIDI y controlarlos mediante el software del ordenador

Los problemas del programa

El programa que ofrecemos está diseñado para demostrar algunos de los principios de la utilización de interrupciones no periódicas con el sistema operativo Amstrad. Es interesante observar algunas de las dificultades del empleo de tal método. Si usted ejecuta el programa y envía datos desde un teclado MIDI, encontrará que ocasionalmente el programa terminará debido a una extralimitación del ACIA, es decir, que se reciba un segundo byte antes de que el byte anterior haya sido borrado del registro de recepción de datos. Esta particularidad obedece a la forma en que el sistema operativo manipula las interrupciones.

La respuesta de interrupciones del Amstrad utiliza el registro alternativo del Z80, establecido temporalmente para almacenar el estado de los registros CPU cada vez que se produce una interrupción. Puesto que sólo está disponible un conjunto de registros alternativos, no se toleran interrupciones de nivel múltiple. Esto significa, a su vez, que las interrupciones se deben inhabilitar a través de la rutina de servicio de interrupciones. Además, debido a que la rutina de servicio incluye varias llamadas para soportar la estructura de eventos del firmware, las interrupciones se podrían inhabilitar durante más del período mínimo de 430 μs crucial, lo que probablemente provocaría la pérdida de uno o más bytes.

Una solución correcta a este problema de extralimitación supondría la sustitución del código manipulador de interrupciones por nuestro propio código introducido en el flujo. Éste, efectivamente, ignoraría las interrupciones periódicas y limitaría seriamente el uso que pudiéramos hacer de las funciones del sistema en firmware.



Cargador hexa:

```
20 org=&408E:loadptr=0
30 WHILE bandera=0
40 GOSUB 1000:IF bandera=1 THEN 60:REM salir
50 POKE org + loadptr,entrada:loadptr=loadptr + 1
60 WEND
70 END
1000 REM **** s/r toma un byte hexa ****
1020 bandera=0 LINE INOUT entrada$
1025 IF entrada$="x" THEN bandera=1:RETURN
1030 IF LEN(entrada$) <> 2 THEN PRINT"error":
     GOTO 1020
1050 entrada=VAL("&"+entrada$):RETURN
```

Si no se posee ensamblador, el programa en código máquina se puede cargar utilizando el cargador hexa en BASIC que ofrecemos aquí. Los bytes desde la columna izquierda se deben entrar de uno en uno con el cargador hexa en ejecución, y luego guardarlos en disco o en cinta usando la instrucción SAVE'NOMBREARCHIVO',B, &4000,&199. Observe que cada byte debe ir seguido por un retorno de carro; de modo que, p. ej., los tres bytes de la segunda línea, 113B00, se deben entrar como 11 <CR> 3B <CR> 00 <CR>

Programa MIDI Amstrad

```
                    org     #4000
          klev:     defs    2
          rxevbk:   defs    140       ;bloque eventos+lectura/escritura
          cont:     equ     #f8e0
          txreg:    equ     #f9e0
          stat:     equ     #fae0
          rxreg:    equ     #fbe0
          ;direcciones de llamada os
          kmrdch:   equ     #bb09
          kmarbr:   equ     #bb45     ;prepara rupturas
          txtout:   equ     #bb5a
          klinev:   equ     #bcef
          kleven:   equ     #bcf2
          kldisa:   equ     #bd0a     ;anula evento asincr
          klpoll:   equ     #b921     ;lleva evento retornos
          klnext:   equ     #bcfb     ;toma siguiente evento sincr
          kldosy:   equ     #bcfe     ;hace evento sincr
          kldone:   equ     #bd01     ;evento sinc hecho
          klsres:   equ     #bcf5     ;limpia cola eventos
          start:                      ;parchea rutina interrupción

F3                  di
113B00              ld      de,#003b
211441              ld      h1,jpnist
010300              ld      bc,3
EDB0                ldir
FB                  ei
          ;inicializa evento recibir
210240              ld      hl,rxevbk
115941              ld      de,rxevrt ;dirección rutina evento
0601                ld      b,1
CDEFBC              call    klinev
3600                ld      (hl),0    ;inicializa lectura/escritura
23                  inc     hl
3600                ld      (h1),0
          ;guardar rutina dir. evento kl
ED5BF3BC            ld      de,(kleven+1)
CBBA                res     7,d
CBB2                res     6,d       ;suprimir bits seleccion rom
ED530040            ld      (klev),de
          ;inicializar 6850
01E0F8              ld      bc,cont
3E03                ld      a,3
ED79                out     (c),a
3E96                ld      a,#96
ED79                out     (c),a
          espera:
CDFBBC              call    klnext
DC0941              call    c,dosync  ;hacer evento si espera
210A40              ld      hl,rxevbk+8
CB7E                bit     7,(hl)
2005                jr      nz,waitl  ;salir si tampón lleno
CD09BB              call    kmrdch
30EE                jr      nc,wait   ;saltar si ninguna tecla pulsada
          espera 1:
CB76                bit     6,(hl)
280C                jr      z,wait2   ;saltar mensaje si no extralim.
21F340              ld      hl,string
0616                ld      b,strlen
7E        nexc:     ld      a,(hl)
CD5ABB              call    txtout
23                  inc     hl
10F9                djnz    nexc
          espera 2:
CDF5BC              call    klsres
F3                  di
3EC9                ld      a,#c9         ;retiene instrucción
323B00              ld      (# 003b),a
3E16                ld      a,# 16
ED79                out     (c),a         ;desactiva interrupciones 6850
FB                  ei
C9                  ret                   ;salida a editor
0A0D      string:   defb    #0A,#0d
41434941            defm    "error
          extralim. ACIA"
0A0D                defb    #0a,#0D
                    strlen: equ $-string
          dosync:                         ;efectúa evento sincr
E5                  push    hl            ;guarda dirección evento
F5                  push    af            ;guarda prioridad anterior
CDFEBC              call    kldosy        ;efectúa evento
F1                  pop     af
E1                  pop     hl
CD01BD              call    kldone
C9                  ret
C31741    jpinst:   jp      rcvint
          rcvint:                         ;rutina interrupción rcv
3A0440              ld      a,(rxevbk+2)
B7                  or      a             ;comprobar cuenta=0
2006                jr      nz,rcvl
320940              ld      (rxevbk+7),a  ;inicializar índices
320A40              ld      (rxevbk+8),a
C5        rcv1:     push    bc            ;guarda estado rom antiguo
210A40              ld      h1,rxevbk+8   ;escribe dirección índice
CB91                res     2,c           ;habilita rom inferior
ED49                out     (c),c
01E0FA              ld      bc,stat
ED78                in      a,(c)
CB6F                bit     5,a
280A                jr      z,rcv15
36FF                ld      (h1),#ff      ;establece bandera extralim.
3E16                ld      a,#16
05                  dec     b
05                  dec     b
ED79                out     (c),a
1817                jr      rcv2
04        rcv15:    inc     b
ED78                in      a,(c)
34                  inc     (hl)          ;incrementa índice
5E                  ld      e,(hl)
CB7B                bit     7,e           ;comprueba si tampón lleno
200E                jr      nz,rcv2       ;sale si lleno
1600                ld      d,0
19                  add     hl,de         ;toma dirección escritura
77                  ld      (hl),a
210240              ld      hl,rxevbk
ED5B0040            ld      de,(klev)
CD1600              call    #0016
C1        rcv2:     pop     bc
ED49                out     (c),c
C9                  ret
          rxevrt:                         ;rutina evento rx hl=par
210940              ld      hl,rxevbk+7
34                  inc     (hl)          ;incrementa índice
5E                  ld      e,(hl)
1C                  inc     e
1600                ld      d,0
19                  add     hl,de         ;apunta a siguiente dato lectura
7E                  ld      a,(hl)
FEF8                cp      # f8
D0                  ret     nc
B7                  or      a
F27641              jp      p,evrt1
47                  ld      b,a
3E0A                ld      a,#0a
CD5ABB              call    txtout
3E0D                ld      a,#0d
CD5ABB              call    txtout
78                  ld      a,b
          evrt1:
CD7A41              call    hexpr         ;imprime car
C9                  ret
                    imphex:               ;impresión hexadecimal
0602                ld      b,2
                    imphex1:
3E00                ld      a,0
ED6F                rld                   ;toma dígito mayor
CD9141              call    digasc        ;convierte a ascii
CD5ABB              call    txtout        ;imprime
10F4                djnz    hexprl
3E20                ld      a," "
CD5ABB              call    toxtout
CD5ABB              call    txtout
C9                  ret
          digasc:                         ;dígito hexa a ascii
C630                add     a,"0"
FE3A                cp      ":"
D8                  ret     c
C607                add     a,"A"-":"
C9                  ret
```

Preestreno sorpresa

Antes de realizar el listado final nos ocuparemos de los procedimientos de clasificación

La utilización de árboles con estructuras internas irregulares (p. ej., tres bifurcaciones desde un nudo, dos desde otro, cuatro desde un tercero, etc.) plantea problemas especiales. La dificultad principal es saber si hemos llegado o no a un nudo terminal. Como recordará, cuando entramos el árbol de manipulación de objetos pudimos, en virtud de su coherencia interna, numerar los nudos por un orden especial: primero los nudos de elección; luego los nudos que saltaban fuera del árbol y después otra vez a él; en tercer lugar, los nudos terminales que dan por resultado una acción o un mensaje; y, por último, los nudos terminales que se limitaban a retornar sin que se emprendiera ninguna acción. Cuando seleccionábamos a lo largo del árbol (en las líneas 5030-5090), simplemente saltábamos a una rutina determinada por el número del nudo. Lamentablemente, si observa los árboles de las páginas 2104 y 2105 e intenta aplicarles este sistema, verá que no es posible.

Lo que se precisa es un sistema infalible para clasificar un árbol que se pueda aplicar a *cualquier* árbol, con independencia de su estructura interna.

Ahora vamos a resolver este problema y a entrar en el programa un árbol del trama. Veamos cómo lo hacemos...

En primer lugar, necesitamos inicializar las variables para la trama que aún no se hayan utilizado. Suprima la línea 190 de su programa y añada ésta:

```
190 DIM t(5,25,4),k(3,30),c(25),s(6),
    h(6):z=0
```

Observará que ahora hemos aplicado la matriz t para hacer frente a nuestros nuevos árboles, y que también hemos añadido algo de espacio en la matriz c para las nuevas condiciones que comprobaremos. Las dos matrices s y h registran, respectivamente, si un personaje ha visto o no un cadáver o manipulado la lata vacía de alimento para gatos. La variable z indica si se ha producido alguna muerte, y se establecerá cuando sea adecuado para indicar el número de personaje de la víctima.

Nosotros programaremos nuestro árbol del siguiente modo. Los cuatro elementos de dimensiones inferior de la matriz t se utilizarán para retener información acerca de cada nudo del árbol, tal como se indica en la tabla de tipos de nudos. Luego, al clasificar a lo largo del árbol, será muy sencillo comprobar el tipo de nudo de cada uno y saltar a la rutina apropiada. Los datos de nudos se almacenan en el listado 1, que es el que debe entrar primero. Después entre las dos líneas siguientes para leer estos datos en la matriz t:

```
230 REM arbol de la trama
240 FOR n=1 TO 22: FOR s=1 TO 4: READ t(2,n,s):
    NEXT s: READ a$: NEXT n
```

Observe los espacios en blanco en las líneas 6270 y 6280. Éstos se añaden para facilitar la lectura, de modo que usted pueda ver con tan sólo una mirada cada grupo de valores de nudos. Los espacios en blanco se leen en la "variable de desperdicios" a$ en la línea 240.

Ahora necesitamos una rutina para "clasificar" este árbol, y saltar a las rutinas correspondientes. Ésta está retenida en las líneas 5400-5550 (listado 2), que es el que usted ha de entrar ahora. El proceso es muy directo. La línea 5470 comprueba el tipo de nudo y le suma uno para generar un número entre 1 y 7, que entonces da como resultado un salto a una de las líneas indicadas. Un nudo de elección simple con sólo dos bifurcaciones (tipo 0) saltará a la 5480, donde se utiliza el valor de la condición retenida en t(2,numero de nudo,2) para decidir si saltar al nudo retenido en t(2,numero de nudo,3) o a t(2,numero de nudo,4).

Cada uno de los restantes tipos de nudo saltan a sus propias líneas y extraen de la matriz t los parámetros correspondientes. Observe que los tipos de nudo 1 y 2 se vectorizan mediante un bloque de salto retenido en las líneas 4500-4570. Esto es para evitar tener que obstruir la rutina de clasificación de árboles con gran cantidad de números de línea distintos cuando vayamos añadiéndole árboles al programa.

Si se remite al diagrama del árbol de la trama, verá que es necesario comprobar algunas condiciones nuevas que todavía no hemos almacenado en la matriz c. Para añadirlas, entre la siguiente línea:

```
2450 c(13)=ABS(z=c): c(14)=ABS(g=C):
     c(15)=ABS(z=0): c(16)=ABS(s(c)=255):
     c(17)=ABS(FNc(z,2)=FNc(c,2)):
     c(18)=ABS(h(c)=255): c(19)=ABS(i=2)
```

Ahora entre los listados 3, 4 y 5. Ya estamos casi listos para probar esta parte del programa. Todo lo que se requiere es variar primero el bucle de control de la línea de modo que rece así:

```
500 REM
510 REM prueba bucle programa
520 REM
530 GOSUB 2100:GOSUB 2150:GOSUB 2240:PRINT:
    PRINT: GOSUB 1000: GOTO 530
```

Probabilidades

Nudo	Tipo	Datos retenidos en t(n.° árbol, n.° nudo,1-4) 1	2	3	4
0	Nudo de elección	0	n.° de condición	nudo al cual saltar si FALSO	VERD.
1	Nudos GOSUB (salir del árbol y después retornar)	1	0	nudo al cual retornar	n.° de rutina
2	Nudo de acción (nudo terminal, salta a rutina)	2	0	n.° de rutina	n.° mensaje, si hubiera
3	Nudo de mensaje (nudo terminal, imprime mensaje)	3	0	0	n.° de mensaje
4	Nudo terminal (salir del árbol, no emprender ninguna acción)	4	0	0	0
5	Nudo aleatorio (generar desplazamiento aleatorio y sumarlo al nudo base)	5	0	nudo base	desplazam. máximo
6	Múltiple opción (sumar valor de condición a nudo base)	6	condición	nudo base	0

Ahora suprima las líneas 540-820, que entramos cuando probamos nuestro árbol de manipulación de objetos, y digite el listado 6. Este listado comprueba las banderas "manipular" y "mover", inicializa condiciones y llama a cada árbol de a uno en las líneas 1200 y 1210. Ahora ya puede ejecutar el programa. En esta etapa se permite a los personajes desplazarse por las dependencias del *pub*, manipular objetos, morirse, e incluso incluso resolver el misterio de las empanadas envenenadas.

Listado 1

Estas líneas retienen los datos para los nudos. Se han incluido espacios ficticios para facilitar la legibilidad.

```
6240 REM
6250 REM datos arbol trama
6260 REM
6270 DATA 0,13,2,22," ",0,14,5,3," ",0,15,21,4," ",
     5,0,18,3," ",0,16,6,7," ",0,17,7,11,"
     ",0,18,9,8," ",0,16,12,13," ",0,19,10,17,"
     ",5,0,14,3," ",1,0,7,1," ",4,0,0,0," ",2,0,1,0,"
     ",2,0,5,1," ",4,0,0,0," ",4,0,0,0," ",2,0,2,4,"
     ",2,0,3,2," ",2,0,4,3
6280 DATA " ",2,0,4,0," ",4,0,0,0," ",4,0,0,0," "
```

Listado 2

Estas líneas clasifican a lo largo de tres árboles, comprobando el tipo de cada nudo y bifurcándose en consecuencia

```
5440 REM
5450 REM clasifica arboles
5460 n=1
5470 ON (t(t,n,1)+1)GOTO 5480,5490,5500,
     5520,5530,5540,5550
5480 k=c(t(t,n,2))+1: n=t(t,n,2+k): GOTO 5470
5490 GOSUB 4530: n=t(t,n,3): GOTO 5470
5500 GOSUB 4570
5510 RETURN
5520 GOSUB 4680: GOSUB 4630: GOSUB 4720
5530 RETURN
5540 a=t(t,n,4): GOSUB 4350: n=t(t,n,3)+ v:GOTO
     5470
5550 k=c(t(t,n,2)): n=t(t,n,3)+k:GOTO 5470
```

Puestos de acción
Las líneas 2810-2920 se encargan de trasladar a los personajes de una habitación a otra. Las líneas 3000-3110 forman la primera parte de la tabla de acciones. Esta tabla contiene las rutinas llamadas por los nudos tipo 2 mediante el bloque de saltos de las líneas 4510-4570. Las rutinas para los "nudos GOSUB" (tipo 1) empiezan en la línea 3900

Listado 3

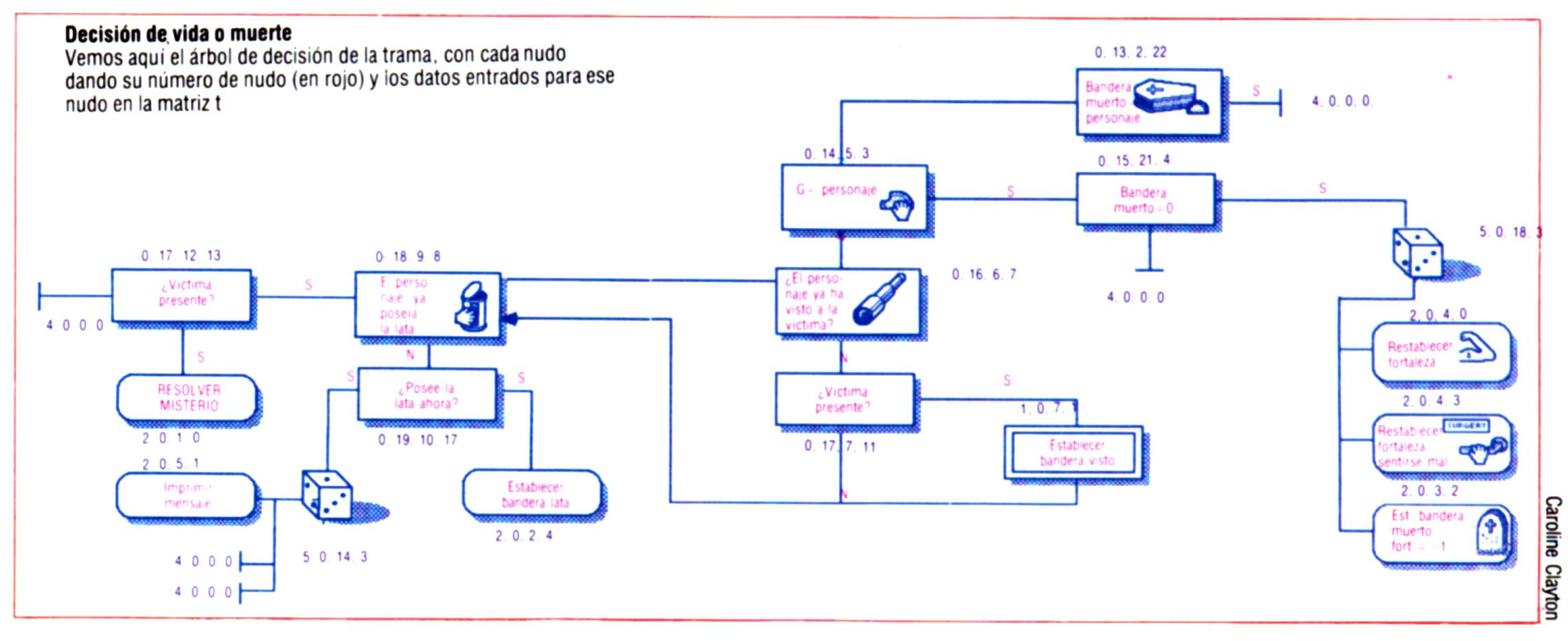
```
2810 REM
2820 REM desplaza un personaje
2830 REM
2840 IF FNc(c,4)<1 THEN RETURN: REM demasiado debil
     para moverse
2850 y=0: f=0: FOR w=2 TO 5: IF I$(VAL(c$(c,2)),w)="0"
     THEN GOTO 2910
2860 GOSUB 4180: IF q=1 THEN f=1: GOTO 2880
2870 GOTO 2910
2880 IF FNc(c,2)=r THEN PRINT c$(c,1);" sale de la
     habitación...": :y=1
2890 c$(c,2)=1$(VAL(c$(c,2)),w): w=5: IF FNc(c,2)=r
     THEN PRINT c$(c,1); " entra en la habitación...";
     :y=1
2900 IF y=1 THEN y=c: GOSUB 2250: c=y: PRINT:
     PRINT: REM actualizar mensaje personajes
     presentes
2910 NEXT w: IF f=0 GOTO 2850
2920 RETURN
3000 REM
3010 REM tabla de acciones
3020 REM
```

Rutinas útiles
Esta parte del programa añade algunas rutinas de bajo nivel nuevas, para generar números aleatorios variables e imprimir mensajes

Listado 4

```
4270 REM
4280 REM imprimir su
4290 REM
4330 REM rutina numeros aleatorios variables
4340 REM
4350 v=INT(RND(2)* a):RETURN
4360 REM
4370 REM imprimir el
4380 REM
4510 REM bloque de saltos
4520 REM
4530 REM
4540 ON t(t,n,4) GOSUB
4550 RETURN
```

Decisión de vida o muerte
Vemos aquí el árbol de decisión de la trama, con cada nudo dando su número de nudo (en rojo) y los datos entrados para ese nudo en la matriz t



Caroline Clayton

```
3030 FOR n=1 TO 3:GOSUB 4090: NEXT n: m$=c$(c,1)+"
     vuelve a mirar el cuerpo y de pronto comprende la
     espantosa verdad. "+CHR$(34)+ "La empanada esta
     elaborada con alimento para gatos"+CHR$ (34)+ "
     ":GOSUB 4630
3040 GOSUB 4390: m$=m$+ "grita, y en una loca carrera
     los clientes reunidos se arremolinan sobre la barra y
     atacan a Fred el barman, quien les suplica en vano que
     tengan piedad de su miserable vida.": GOSUB
     4630:GOSUB 4720
3050 FOR n=1 TO 2000:NEXT n: GOSUB 4720: m$="...y al
     dia siguiente a Fred el barman no se lo ve por ninguna
     parte. No obstante, hay muchisimas empanadas de
     extraña forma para alimentar a la siempre famelica
     clientela del Dog and Bucket...": GOSUB 4630:GOSUB
     4720: END
3060 h(c)=255:GOSUB 4680: m$=c$(c,1)+ m$: GOSUB
     4630: GOSUB 4720: RETURN
3070 z=c: GOSUB 4680: n$=c$(c,1)+m$:GOSUB
     4300:m$=n$+m$+ "estomago, e inmediatamente
     muere. Los otros personajes estan demasiado
     concentrados en sus bebidas como para darse
     cuenta...": GOSUB 4630: GOSUB 4720: g=0:
     c$(c,4)="-1": RETURN
3080 c$(c,4)="10": g=0: IF t(t,n,4) > 0 THEN GOSUB
     4680:m$=c$(c,1)+m$:GOSUB 4630: GOSUB
     4720
3090 RETURN
3100 a=20: GOSUB 4350: IF v<>5 THEN RETURN
3110 GOSUB 4680: GOSUB 4630: GOSUB 4720:
     RETURN
3900 REM
3910 REM gosub tabla
3920 REM
3930 s(c)=255: m$=c$(c,1)+ "se arrodilla junto al cuerpo
     postrado de "+c$(z,1)+". La horrible verdad toma
     cuerpo lentamente, pero los demas parecen estar
     demasiado borrachos como para prestarle ninguna
     atencion inmediata...": GOSUB 4630: GOSUB 4720:
     RETURN
```

```
4560 REM nudos de accion
4570 ON t(t,n,3) GOSUB 3030,3060,3070,3080,3100:
     RETURN
4600 REM
4610 REM imprimir mensajes si el jugador esta presente
4620 REM
4630 IF FNc(c,2)=r THEN PRINT m$;
4640 m$="": RETURN
4650 REM
4660 REM seleccionar un mensaje de la sentencia de
     datos
4670 REM
4680 RESTORE 7030: FOR m=1 TO t(t,n,4):READ m$: NEXT
     m: RETURN
4690 REM
4700 REM imprimir una linea en blanco
4710 REM
4720 IN FNc(c,2)=r THEN PRINT
4730 RETURN
```

Listado 5

Recibir el mensaje

La línea 4680 emplea líneas DATA para recuperar mensajes. Lamentablemente, los ordenadores Commodore no pueden RESTORE a un número de línea, de modo que se ha de utilizar una técnica distinta; remítase a los correspondientes complementos que ofreceremos en el próximo capítulo

```
7000 REM
7010 REM datos mensajes
7020 REM
7030 DATA "Un extraño olor invade el aire... ¿podria ser el
     aroma de Catty-Kit A La Carte?", "de pronto se
     desploma sobre el suelo, apretandose el estomago",
     "parece muy enfermo, y advierte a los demas que no
     toquen la empanada."
7040 DATA "examina atentamente la lata, y su semblante
     adquiere un aire pensativo."
```

Complementos al BASIC

Tal como está impreso, el listado se ejecutará sin ninguna modificación en la gama de ordenadores Amstrad. En el próximo capítulo incluiremos complementos para otras máquinas

Listado 6

Dirigiendo el reparto

El manipulador de personajes procesa a cada personaje de uno en uno. Estas líneas comprueban las banderas «mover» y «manipular» y llaman a las rutinas adecuadas para trasladar personajes o clasificar árboles. Se han incluido algunas sentencias REM para facilitar la lectura

```
1000 REM
1010 REM manipulador personajes
1020 REM
1030 REM comprueba si se ha pulsado alguna tecla
1040 GOSUB 4260: IF i$<> "" THEN GOSUB 2040: RETURN
1050      REM procesa cada personaje por turno
1060 FOR c=1 TO 6
1070      REM comprueba 'bandera manipular'
1080 IF FNc(c,10)>0 THEN c$(c,10)=FNm$(c$(c,10),1):
     GOTO 1500
1090      REM bandera=0 para restaurar bandera y procesar
          personaje
1100 RESTORE: FOR n=1 TO c*10+c-1: READ c$(c,10):
     NEXT n
1110 IF FNc(c,10)=0 THEN GOTO 1500: REM valor por
     defecto=0 por tanto no procesa
1120      REM comprueba bandera mover
1130 IF FNc(c,11)> 0 THEN c$(c,11)=FNm$(c$(c,11),1):
     GOTO 1190
1140      REM bandera mover=0 por tanto restaura bandera y
          mueve personaje
1150 RESTORE: FOR n=1 TO c*11: READ c$(c,11): NEXT n
1160 IF c$(c,11)="0" THEN GOTO 1180
1170 GOSUB 2840: GOTO 1500
1180      REM clasificar a lo largo de los arboles
1190 GOSUB 2400: REM inicializa condiciones
1200 t=2:GOSUB 5460: REM arbol trama
1210 IF FNc(c,4) >0 THEN GOSUB 5000: REM arbol
     manipulacion objetos
1500 NEXT c: GOTO 1030: REM hacer siguiente personaje -
     cuando todos terminados volver a comprobar si tecla
     pulsada/hacerlo otra vez
```





Los designios del azar

Es imprescindible recurrir al cálculo de probabilidades para dar un enfoque científico a los juegos de azar. Con este fin, implementaremos la regla de Bayes

Las apuestas se basan totalmente en las probabilidades y, sin embargo, la mayor parte de los apostadores tienen una idea sumamente vaga acerca de la teoría de las probabilidades. Ésta es, posiblemente la razón por la cual continúan perdiendo dinero. El tema de las probabilidades está lleno de paradojas y trampas ocultas, de modo que no es sorprendente que nos resulte difícil enunciar una teoría que sea precisa acerca de la incertidumbre. El apostador puede llegar muy lejos con la ayuda de un micro, un poco de autodisciplina y cierto conocimiento de la teoría elemental de las probabilidades.

Lo primero que debe saber el apostador bien informado es cómo convertir las posibilidades en probabilidades y viceversa, a pesar de lo cual es sorprendente el escaso número de ellos que así lo hacen. Existen dos clases de posibilidades: las a favor (F) y las en contra (C). Los corredores de apuestas suelen hacer explícitas las posibilidades de un evento de resultado, como de que un caballo gane una carrera, excepto cuando el evento es lo que se dice "seguro". El cuadro se complica por la práctica usual de citar un par de enteros tales como 6 a 4 para definir las posibilidades, cuando 3 a 2 parece mucho más directo.

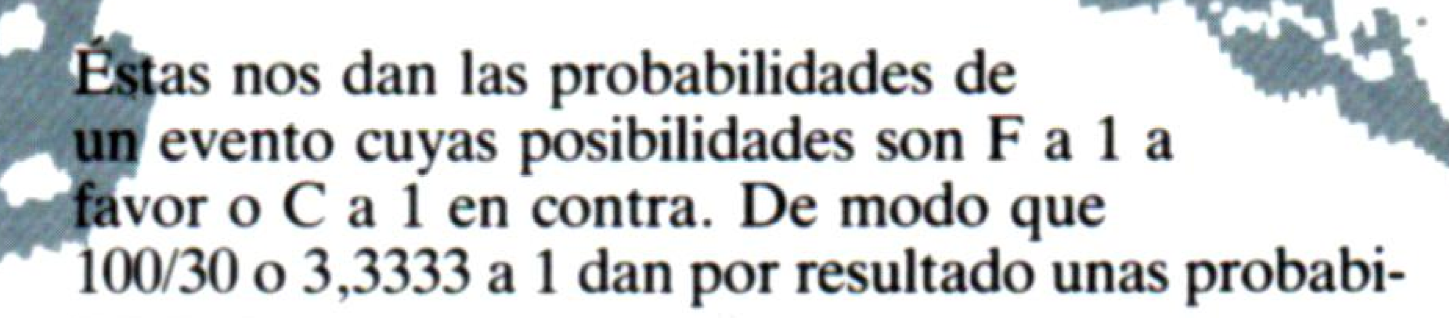
Es una buena idea simplificar las cosas, reduciendo las variedades de posibilidades a una media uniforme. Esto nos será de ayuda cuando lleguemos a los cálculos informatizados. El primer paso para convertir las posibilidades en probabilidades es expresar las posibilidades con un único número. Esto se puede hacer utilizando una de las fórmulas consignadas abajo, donde la c (contra) y f (favor) minúsculas son los dos números citados por el corredor de apuestas.

F=f/c
C=c/f

De modo que si las posibilidades se dan como 100 a 30 en contra, c=100 y f=30. Por tanto, las posibilidades a favor (F) son 30/100, lo que es igual a 0,30, y las posibilidades en contra son 100/30=3,3333. Ahora hemos reducido las posibilidades a una escala común: 7 a 2 es 3,5 a 1, 100 a 30 es 3,3333 a 1, 11 a 4 es 2,75 a 1, y así sucesivamente. Con ello ya se facilitan las comparaciones.

El siguiente paso es transformar las posibilidades en probabilidades, con una de estas fórmulas:

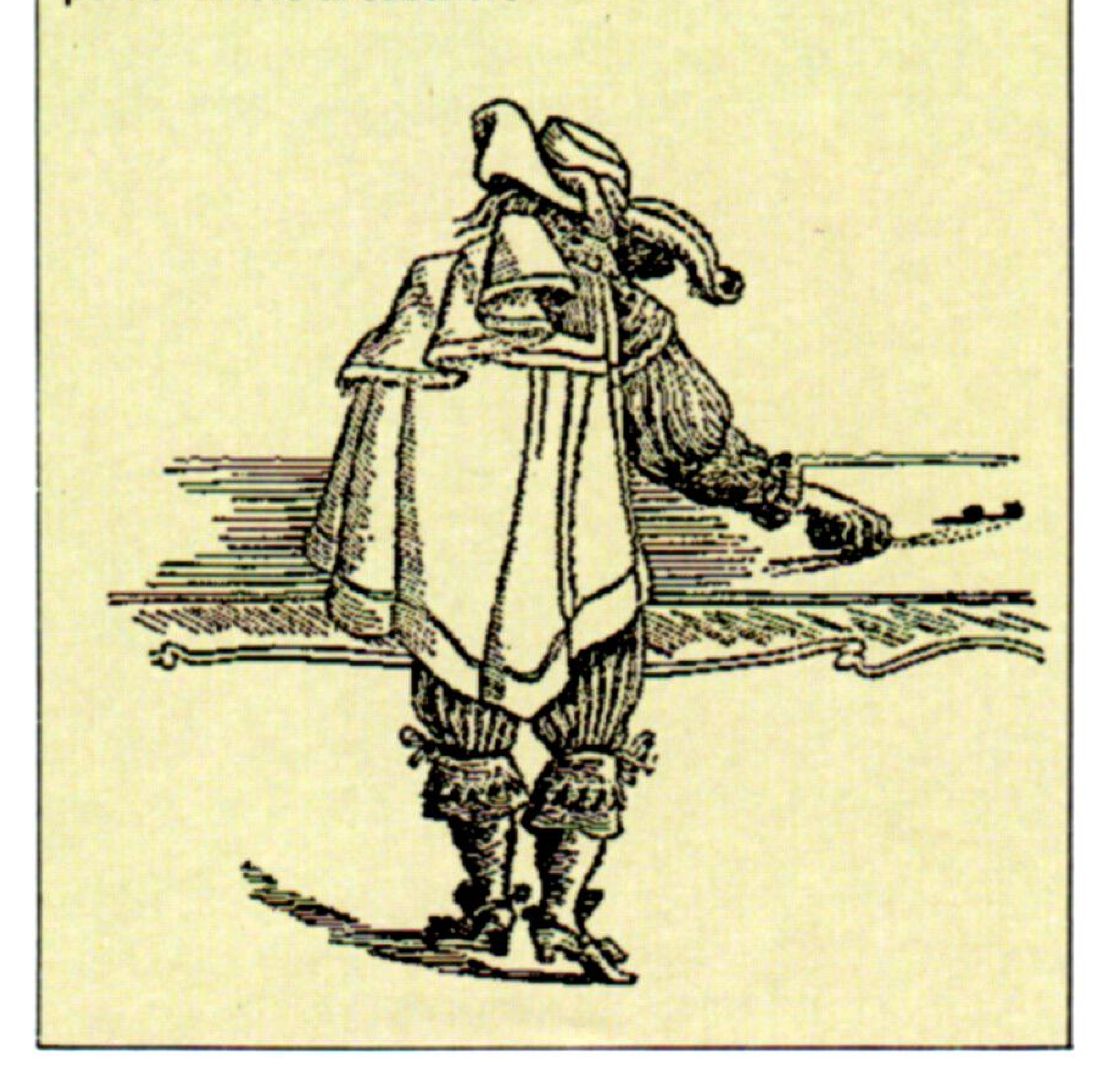
P=F/(F+1)
P=1/(C+1)



Éstas nos dan las probabilidades de un evento cuyas posibilidades son F a 1 a favor o C a 1 en contra. De modo que 100/30 o 3,3333 a 1 dan por resultado unas probabi-

Teoría de las probabilidades

El estudio serio de las probabilidades matemáticas comenzó, como cabría esperar, con una serie de apuestas perdidas. El escritor francés Antoine Gombaud, caballero de Méré (1607-1685), había logrado ganar mucho dinero apostando repetidas veces que él podía, en cuatro tiros de un dado, sacar al menos un seis. Lamentablemente, su éxito con esta apuesta fue origen de una clara retracción de apostadores. Por lo tanto, en 1654 pasó a apostar que en 24 tiradas de un par de dados arrojaría al menos una vez un doble seis. En aquel entonces, el único método de determinar las probabilidades en tales casos consistía en lanzar los dados tantas veces como fuera posible y registrar el resultado. Esto no sólo es tedioso, sino también inexacto, de modo que cuando la nueva apuesta del caballero comenzó a amenazarlo con la bancarrota, le escribió al matemático Blaise Pascal recabando su ayuda. De esta forma las probabilidades comenzaron a estudiarse seriamente. Las posibilidades de sacar un seis en cuatro tiradas son de 14 a 13 a favor. Aplicando la fórmula presentada en este capítulo, ¿podría decir por qué la segunda apuesta le hizo perder dinero al caballero?





lidades de 1/4,3333 = 0,2308. Las probabilidades (P) siempre se expresan dando por sentado que el evento se produzca verdaderamente (como que un caballo gane). Si usted desea las probabilidades de que el evento *no* se produzca (Q), puede usar la fórmula:

$Q=1-P$

Recuerde que $P + Q=1$, las probabilidades de que un evento se produzca más las probabilidades de que ese evento no tenga lugar deben sumar 1. Observe también que las posibilidades elevadas (en contra) están asociadas con eventos improbables y con escasas probabilidades (a favor).

Las posibilidades de los corredores de apuestas son menores que las auténticas posibilidades; por decirlo en otras palabras, hacen que los eventos en cuestión parezcan más probables de lo que son en realidad. Las razones de esto se hacen evidentes cuando analizamos los hechos desde el punto de vista del corredor de apuestas. Éste ha de ganarse la vida y cubrir sus gastos. En consecuencia, crea lo que podría denominarse un registro de apuestas "compensatorio". Podemos ilustrar esto con una carrera de caballos imaginaria:

Corredor	Apuestas	Posib.	Riesgo
Apricot	10 000 pts.	5-6	18 333 pts.
Olivetti	7 500 pts.	13-8	19 687 pts.
Alice	2 000 pts.	8-1	18 000 pts.
Apple	500 pts.	33-1	17 000 pts.
	20 000 pts.		

En este caso, se han apostado 20 000 pts, la mitad de esta cifra a *Apricot* y el resto tal como se indica. Las posibilidades se han escogido como para mantener el riesgo del corredor de apuestas por debajo del total apostado, es decir, 20 000 pts, suceda lo que suceda. En consecuencia, si gana *Alice*, los apostadores que lo respaldaron por la cantidad de 2 000 pts, recuperarán sus apuestas, más ocho veces lo que apostaron (16 000 pts) en concepto de beneficios, lo que totalizan 18 000 pts. Esto deja un margen de beneficio de 2 000 pts (el 10 %) para el corredor de apuestas. En un registro de apuestas imparcial, *Apricot* sería dinero a la par y *Alice* par-



Probabilidades posibles

Trabajemos las posibilidades de la conversión de probabilidades. Abajo vemos las posibilidades citadas en un cupón de apuestas para un partido de fútbol entre equipos ingleses: Leicester contra Everton.

15/8 Leicester 5/2 Everton 1/1

Esto indica que las posibilidades en contra de que el equipo local (Leicester) gane son de 15 a 8; las posibilidades de un empate son de 5 a 2, y las posibilidades de que el Everton gane fuera de casa son de 1 a 1, o a la par. (Este partido se jugó en el verano de 1985: Everton se anticipó en el marcador, pero al final ganó el Leicester por tres goles a uno, ¡de modo que los perdedores no siempre pierden!) En primer lugar, normalicemos las posibilidades: C=c/f.

Local	15/8	= 1,875
Empate	5/2	= 2,50
Visitante	1/1	= 1,0

Expresémoslas en probabilidades: P=1/(C+1).

Local	1/2,875	= 0,3478
Empate	1/3,5	= 0,2857
Visitante	1/2	= 0,5000
Total		= 1,1335

Observen que estas probabilidades suman más de uno: 1,1335, para ser exactos. Esto viola las leyes de las probabilidades, pero no es ilegal. Significa que en este partido el margen de los corredores de apuestas es del 13,35 %. Si recuerda que las posibilidades elevadas (en contra) se traducen en escasas posibilidades (a favor), verá que los corredores de apuestas reducen las posibilidades para obtener su porcentaje. Recuerde que las posibilidades de los corredores tienden a ser menores que las verdaderas posibilidades. Para ganar dinero tendrá que encontrar casos en los que no se hayan reducido demasiado.

Para ahorrarle parte del trabajo con lápiz y papel (y con la mente también), nuestro listado elimina la labor monótona de estos cálculos. También calcula el margen de los corredores de apuestas. Veamos dos muestras, una con el partido que mencionamos arriba y la segunda con el Derby de 1985 (con precios de partida) que se corrió en Epsom

```
 10 REM ******************************
 15 REM * CALCULOS DE PROBABILIDADES *
 20 REM ******************************
100 PRINT'"Calculador de probabilidades:"
110 PRINT "Por favor de las posibilidades como dos numeros;"
120 PRINT "por ej. 3.1 para 3 a 1 etc."
130 PRINT "Use 0,0 para terminar una entrada."'
140 at%=&0102040A: REM formato
150 N=0
160 TP=0: TA=0
190 REM -- Bucle principal:
200 REPEAT
202   @%=4
210   N=N+1
220   PRINT "Corredor ";N;" su nombre es";
222   INPUT NOMBRE$
225   PRINT "Las posibilidades en contra son ";
230   INPUT C,F
240   IF C< =0 AND F<=0 THEN GOTO 310: REM salir
250   P=F/(F+C)
260   C=C/F
270   TP=TP+P: REM prob.totales
280   TC=TC+C: REM pos. totales
290   PRINT "Para el corredor no. ";N;", ",NOMBRE$
295   @%=at%
296   PRINT "las probabilidades son: ";P
300   PRINT
310   UNTIL C< =0 OR F< =0
320 N=N-1
322 @ %=at%
330 PRINT
333 PRINT "El importe en juego es ";100/TP;"%"
335 PRINT "Margen de ganancia de corredores:
    ";100*(TP-1);"%"
360 IF TP<1 THEN PRINT "Estás bromeando!";CHR$7
999 END
>RUN
```



tiría con 9 a 1. Pero ¿a quién le interesaría llevar un registro de apuestas imparcial?

Una particularidad de nuestro registro de apuestas imaginario que responde a la realidad es que a los *outsiders* se les dan posibilidades relativamente escasas. Ello significa que al corredor de apuestas por lo general le va mejor cuando gana un *long shot*, de modo que tenga cuidado al respaldar "caballos oscuros". Es probable que usted esté subvencionando opciones más unánimes. Esto se debe a que los apostadores se muestran reacios a ir muy por debajo del dinero a la par, de modo que los corredores se ven limitados en cuanto a las posibilidades que pueden ofrecer para los favoritos claros. Y lo compensan a costa de los que "no ofrecen esperanzas".

De cuando en cuando los corredores de apuestas no consiguen cubrir sus riesgos, ya sea porque reaccionen con demasiada lentitud a los cambios repentinos en las apuestas, o bien porque abren con cotizaciones no realistas, aunque tales situaciones son raras. No se quede con la impresión de que puede ganar a los corredores, porque no puede hacerlo. Lo que sí puede lograr es ganar algún dinero a sus compañeros apostadores, con el corredor actuando como intermediario y quedándose con su "tajada".



Jugando a las quinielas
Los apostadores rellenan las quinielas de muchas maneras, pero uno de los métodos más populares es el de 8 resultados fijos sobre 10, en el que el apostante coloca cruces junto a diez partidos. Conocidos los resultados, se toman las ocho mejores combinaciones para maximizar la cantidad de aciertos. Los métodos para seleccionar los partidos varían mucho, pero la posibilidad de haber escogido al azar ocho empates es sumamente remota



```
Calculador de probabilidades:
Por favor de las posibilidades como dos numeros;
por ej. 3,1 para 3 a 1 etc.
Use 0,0 para terminar una entrada.

Corredor 1 su nombre es ?local
Las posibilidades en contra son ?15,8
Para el corredor no. 1, local
la probabilidad es de: 0,3478

Corredor 2 su nombre es ?empate
Las posibilidades en contra son ?5,2
Para el corredor no. 2, empate
la probabilidad es de: 0,2857

Corredor 3 su nombre es ?visitante
Las posibilidades en contra son ?1,1
Para el corredor no. 3, visitante
la probabilidad es de: 0,5000

Corredor 4 su nombre es?
Las posibilidades en contra son ?0,0

El importe en juego es 88,2192%
Margen de ganancia de los corredores:
13,3540%
>
>
>RUN


Calculador de probabilidades:
Por favor de las posibilidades como dos numeros;
por ej. 3,1 para 3 a 1 etc.
Use 0,0 para terminar una entrada.

Corredor 1 su nombre es ?Slip Anchor
Las posibilidades en contra son ?9,4
Para el corredor no. 1, Slip Anchor
la probabilidad es de: 0,3077

Corredor 2 su nombre es ?Law Society
Las posibilidades en contra son ?5,1
Para el corredor no. 2, Law Society
la probabilidad es de: 0,1667

Corredor 3 su nombre es ?Damister
Las posibilidades en contra son ?16,1
Para el corredor no. 3, Damister
la probabilidad es de: 0,0588

Corredor 4 su nombre es ?Supreme Leader
Las posibilidades en contra son ?10,1
Para el corredor no. 4, Supreme Leader
la probabilidad es de: 0,0909
```



```
Corredor 5 su nombre es ?Lanfranco
Las posibilidades en contra son ?14,1
Para el corredor no. 4, Lanfranco
la probabilidad es de: 0,0667

Corredor 6 su nombre es ?Reach
Las posibilidades en contra son ?33,1
Para el corredor no. 6, Reach
la probabilidad es de: 0,0294

Corredor 7 su nombre es ?Theatrical
Las posibilidades en contra son ?10,1
Para el corredor no. 7, Theatrical
la probabilidad es de: 0,0909

Corredor 8 su nombre es ?Phardante
Las posibilidades en contra son ?40,1
Para el corredor no. 8, Phardante
la probabilidad es de: 0,0244

Corredor 9 su nombre es ?Royal Harmony
Las posibilidades en contra son ?40,1
Para el corredor no. 9, Royal Harmony
la probabilidad es de: 0,0244

Corredor 10 su nombre es ?Snow Plant
Las posibilidades en contra son ?100,1
Para el corredor no. 10, Snow Plant
la probabilidad es de: 0,0099

Corredor 11 su nombre es ?Petowski
Las posibilidades en contra son ?33,1
Para el corredor no. 11, Petowski
la probabilidad es de: 0,0294

Corredor 12 su nombre es ?Seurat
Las posibilidades en contra son ?33,1
Para el corredor no. 12, Seurat
la probabilidad es de: 0,0294

Corredor 13 su nombre es ?Shadeed
Las posibilidades en contra son ?7,2
Para el corredor no. 13, Shadeed
la probabilidad es de: 0,2222

Corredor 14 su nombre es ?Main Reason
Las posibilidades en contra son ?200,1
Para el corredor no. 14, Main Reason
la probabilidad es de: 0,0050

Corredor 15 su nombre es ?
Las posibilidades en contra son ?0,0

El importe en juego es 86,5215%
Margen de ganancia de los corredores:
15,5781%
>
```

Dentro de celdas

Habiendo escrito las rutinas de manipulación de gráficos y de procesamiento de fórmulas para nuestro programa de hoja electrónica, estamos en condiciones de añadir las rutinas que permiten entrar datos y fórmulas en la hoja electrónica y efectuar los cálculos que sean pertinentes

El código que manipula la entrada de fórmulas se halla comprendido entre las líneas 2000 y 2050. Esta sección de código utilizaba una instrucción INPUT de BASIC para tomar una fórmula del usuario. Esta fórmula, por supuesto, está en forma de infijos e inicialmente se almacena en D$. La rutina pasa luego a almacenar la fórmula en la matriz F$(). En esta etapa se borra la entrada correspondiente en la matriz que retiene la versión en notación polaca inversa, PS$().

Las fórmulas para las celdas de A1 a A15 se retienen en F$(1) a F$(15); las celdas de B1 a B15 se retienen en F$(16) a F$(30), y así sucesivamente. El elemento correcto de F$() se puede hallar en cualquier momento a partir de las coordenadas del cursor de la hoja electrónica, usando la fórmula (Y-1)*15+X.

La siguiente rutina, entre las líneas 2100 y 2250, se encarga de entrar los datos en una celda de la hoja. Esta subrutina se llama desde la rutina de exploración del teclado de la línea 1170 si se pulsa una tecla numérica. La rutina imprime un mensaje para indicar que usted está entrando datos en la celda actual y luego toma los datos carácter por carácter en la línea 2120. Si la tecla pulsada es una de las teclas del cursor o la barra espaciadora, entonces la rutina se termina y los datos numéricos que se hayan reunido hasta entonces se colocan en una matriz M(,). Las líneas 2149-2170 comprueban que los datos entrados sean válidos y, si lo fueran, añade el número a E$. Una limitación de nuestro programa de hoja electrónica es que cada celda puede aceptar un máximo de sólo cinco caracteres; por ello se comprueba la entrada extra en la línea 2160 y se imprime un mensaje en la línea 2220.

La rutina de cálculo empieza en la línea 2300 y actúa a modo de bucle de control para las rutinas de traducción a polaca inversa desarrolladas en los dos últimos capítulos, antes de evaluar verdaderamente las fórmulas, en la subrutina de la línea 2370.

La primera rutina va trabajando con la matriz de fórmulas de infijos, F$() y la correspondiente matriz de notación polaca inversa, PS$, traduciendo todas las fórmulas recién entradas a polaca inversa, llamando a la subrutina de la línea 4000. Si en el elemento actual que se está comprobando hay una entrada, entonces se llama a la rutina de evaluación.

La rutina de evaluación se vale de la comprobación de legalidad polaca inversa de la línea 4700, que deposita los elementos de la serie polaca inversa con la que está trabajando en una matriz, GS(). Estos elementos son ya sea operadores (como + y −) o bien operandos (A1, B5, 3, etc.). La rutina de evaluación trabaja luego sobre los elementos de la serie polaca inversa de acuerdo a las siguientes condiciones:

- Si el elemento es una constante (diferenciada de la dirección de celda por el hecho de que el carácter de la izquierda será numérico y no alfabético), entonces se coloca el valor en la matriz C() y su posición en C() se coloca en una pila, ST().
- Si el elemento es una dirección de celda, entonces se halla su valor a partir de la matriz M(,) y se coloca en C(). Su posición se coloca en la pila en C().
- Si el elemento es un operador, entonces se lleva a cabo la operación sobre dos (o, en el caso de un menos unario, uno) operandos previamente apilados. De hecho, la pila retiene las posiciones de los operandos de C() y, por tanto, los números que se toman de la pila se utilizan para localizar los elementos correctos en C(). El resultado se coloca en C() y se suprimen de la pila los dos operandos. Luego se coloca en la pila la posición del resultado en C().

Tras haber procesado la serie completa, el resultado de la evaluación de la serie estará retenido en el último elemento de C() y se podrá transferir al elemento apropiado de M(,), la matriz que retiene todos los valores de celdas.

Complementos al BASIC

Amstrad CPC 464/664:
Introduzca las siguientes modificaciones en la versión para el Commodore 64:

```
2010 LOCATE 1,22
2020 PRINT "NUEVA FORMULA          ";
     CHR$(11)
2103 LOCATE 1,22
2120 A$="":WHILE A$="":A$=INKEY$:WEND
2130 IF A$=CHR$(243) OR A$=CHR$(242) OR
     A$=CHR$(241) OR A$=CHR$(240) THEN
     2250
2180 LOCATE H(X+1-H1)-1,V(Y-V1+1)
2190 PRINT CHR$(24);SPACE$((5);CHR$(11)
2200 LOCATE H(X+1-H1)+4-LEN(E$),V(Y-V1
     +1):PRINT E$;CHR$(24)
2220 LOCATE 1,22
2305 LOCATE 1,22
2315 Q$=CHR$(J+64)+MID$(STR$(I),2,2):
     LOCATE 1,1:PRINT "CELDA:";Q$;""
```

BBC Micro:
Introduzca las siguientes modificaciones en la versión para el Commodore 64:

```
2010 PRINT TAB(0,22);
2103 PRINT TAB(0,22);
2120 A$=get$
2130 IF A$=CHR$(136) OR A$=CHR$(137) OR
     A$=CHR$(138) OR A$=CHR$(139) THEN
     2250
2180 COLOUR 2:COLOUR 129
2190 PRINT TAB(H(X+1-H1)-1,V(Y-V1+1)
     -1);" "
2200 PRINT TAB(H(X+1-H1)+4-LEN(E$),
     V(Y-V1+1)-1);E$
2205 COLOUR 1:COLOUR 128
2220 PRINT TAB(0,22);
2230 I$=GET$
2240 PRINT TAB(0,22);
2305 PRINT TAB(0,22);
2315 Q$=CHR$(J+64)+MID$(STR$(I),2,2):
     PRINT TAB(0,0);"CELDA:";Q$;""
```



Rutinas de entrada y cálculo

Commodore 64:

```
1170 IF A$>="0" AND A$=<"9" THEN GOSUB 2100:REM
     ENTRAR DATOS NUMERICOS
2000 REM *** RUTINA ENTRAR FORMULA ***
2010 GOSUB 1950:REM MOVER CURSOR A LINEA DE
     ENTRADA
2020 PRINT "NUEVA FORMULA:    "CU$
2030 INPUT "NUEVA FORMULA:";D$
2040 LET F$((Y-1)*15+X)=D$:LET PS$((Y-1)*15+X)=""
2050 GOSUB 1900:RETURN
2100 REM *****ENTRAR DATOS EN LA CELDA *****
2103 GOSUB 1950:REM MOVER CURSOR A LA LINEA DE
     ENTRADA
2104 PRINT "                    ENTRANDO -DATOS
2105 LET P=0: LET E$=""
2110 IF A$<>"" THEN 2150
2120 GET A$:IF A$="" THEN 2120
2130 IF A$=CHR$(29) OR A$=CHR$(157) OR A$=CHR$(17)
     OR A$=CHR$(145) THEN 2250
2135 IF A$=" " THEN 2250
2140 IF A$="."THEN 2160
2150 IF A$< "0" OR A$> "9" THEN 2120
2160 LET P=P+1:IF P>5 THEN 2220
2170 LET E$=E$+A$
2180 PRINT CO$;:FOR C=1 TO V(Y+1-H1)-1:PRINT
     CD$;:NEXT C
2190 PRINT TAB(H(X+1-H1)-1);CHR$(18); "          ";
     CU$
2200 PRINT TAB(H(X+1-H1)+4-LEN(E$));CHR$(18);E$
2210 GOTO 2120
2220 GOSUB 1950:REM MOVER CURSOR HASTA LINEA DE
     ENTRADA
2225 PRINT "             ERROR-ENTRADA IGNORADA"
2230 GET I$:IF I$="" THEN 2230
2240 PRINT CO$;:FOR K=1 TO 22:PRINT CD$;:NEXT K
2245 PRINT "                              ":GOTO
     2120
2250 LET M(Y,X)=VAL(E$):GOSUB 1900:RETURN
2300 REM ***** CALCULA LA HOJA *****
2305 GOSUB 1950:REM MOVER CURSOR A LINEA DE
     ENTRADA
2306 PRINT"          CALCULANDO-ESPERE POR FAVOR "
2310 FOR J=1 TO 15:FOR I=1 TO 15
2315 Q$=CHR$(J+64)+MID$(STR$(I),2,2):PRINT
     CO$;CD$;"CELDA:" ;Q$;" "
2320 LET P$=PS$((J-1)*15+I)
2325 IF P$<>"" THEN GOSUB 4700:GOSUB 2370:GOTO
     2350
2330 LET C$=F$((J-1)*15+I)
2335 IF C$="" THEN 2350
2340 GOSUB 4000:GOSUB 2370:REM  DECODIFICA
     FORMULA
2350 NEXT I,J:GOSUB 1700:RETURN
2370 REM ***** EVALUA FORMULA *****
2375 SP=0:FOR K=1 TO 20:ST(SP)=0:NEXT K
2380 FOR K=1 TO CP
2390 LET T$=G$(K)
2400 IF T$="+" THEN 2500
2405 IF T$="-" THEN 2510
2410 IF T$="*" THEN 2520
2415 IF T$="/" THEN 2530
2420 IF T$="^" THEN 2540
2425 IF T$="&" THEN 2550
2430 IF T$="" THEN 2450
2432 TE$=MID$(T$,1,1)
2435 IF (TE$> ="0" AND TE$<="9") OR TE$="."
     THEN C(K)=VAL(T$):GOTO 2445
2440 LET C(K)=M(ASC(MID$(T$,1,1))-64,VAL(MID$
     (T$,2,2)))
2445 SP=SP+1:ST(SP)=K
2450 NEXT K
2460 LET M(J,I)=C(K-1):RETURN
2500 C(K)=C(ST(SP)-1))+C(ST(SP)):ST(SP)=0:SP-1:
     ST(SP)=K:GOTO 2450
2510 C(K)=C(ST(SP)-1))-C(ST(SP)):ST(SP)=0:SP=
     SP-1:ST(SP)=K:GOTO 2450
2520 C(K)=C(ST(SP-1))*C(ST(SP)):ST(SP)=0:SP=
     P-1:ST(SP)=K:GOTO 2450
2530 C(K)=C(ST(SP-1))/C(ST(SP)):ST(SP)=0:SP=
     P-1:ST(SP)=K:GOTO 2450
2540 C(K)=C(ST(SP-1))^C(ST(SP)):ST(SP)=0:SP=
     P-1:ST(SP)=K:GOTO 2450
2550 C(K)=0-C(ST(SP)):ST(SP):=K:GOTO 2450
3010 DIM H(5),V(8),ST(20),ST$(20),E$(20),G$(20),
     C(20)
```

Sinclair Spectrum:

```
2000 REM *******************
2001 REM * ENTRAR FORMULAS *
2002 REM *******************
2010 PRINT AT 18,0;" ENTRAR NUEVA
     FORMULA "
2020 INPUT LINE D$
2030 LET F$((Y-1)*15+X,1 TO LEN D$)=D$:
     LETH$((Y-1)*15+X,1 TO )=""
2050 GO SUB 1900
2060 RETURN
2100 REM ***************************
2101 REM * ENTRAR DATOS EN LA CELDA *
2102 REM ***************************
2105 PRINT AT 18,0;"    ENTRANDO-DATOS "
2110 LET P=0: LET B$=""
2120 LET A$=INKEY$: IF A$="" THEN GO TO
     2120
2130 IF A$=CHR$ (13) THEN GO SUB 1650: GO
     TO 2250
2140 IF A$="." THEN GO TO 2160
2150 IF A$< "0" OR A$>"9" THEN GO TO 2120
2160 LET P=P+1: IF P> 5 THEN GO TO 2220
2170 LET B$=B$+ A$
2180 PRINT AT V(Y+1-V1),H(X+1-H1);" "
2190 PRINT AT V(Y+1-V1),H(X+1-H1)+
     5-LEN B$;B$
2210 GO TO 2120
2220 PRINT AT 18,0;" ERROR-ENTRADA
     IGNORADA "
2230 LET I$=INKEY$: IF I$="" THEN GO TO
     2230
2240 PRINT AT 18,0;"                ":GO TO 2110
2250 LET M(Y,X)=VAL (B$): GO SUB 1900
2260 RETURN
2300 REM *******************
2301 REM * CALCULAR LA HOJA *
2302 REM *******************
2305 PRINT AT 18,0;"  CALCULANDO-ESPERE
     POR FAVOR"
2310 FOR J=1 TO 15: FOR I=1 TO 15
2315 PRINT AT 0,0;"CELDA:";CHR$
     (J+64);STR$ (I); ""
2320 LET P$=H$((J-1)*15+I,1 TO)
2325 IF P$(1)<>" "THEN GO SUB 2355: GO
     SUB 4700: GO SUB 2370: GO TO 2350
2330 LET C$=F$((J-1)*15+I,1 TO)
2335 IF C$(1)=" " THEN GO TO 2350
2340 GO SUB 4000: GO SUB 2370
2350 NEXT I: NEXT J: GO SUB 1700: RETURN
2355 FOR Z=1 TO LEN P$: IF P$(Z)= "" THEN
     GO TO 2357
2356 NEXT Z
2357 LET P$=P$(1 TO Z-1):RETURN
2370 REM **********************
2371 REM * DECODIFICAR FORMULA *
2372 REM **********************
2375 LET SP=0: DIM S(20)
2380 FOR K=1 TO CP
2390 LET T$=G$(K)
2400 IF T$(1)="+" THEN GO TO 2500
2405 IF T$(1)="-" THEN GO TO 2510
2410 IF T$(1)="*" THEN GO TO 2520
2415 IF T$(1)="/" THEN GO TO 2530
2420 IF T$(1)="^" THEN GO TO 2540
2425 IF T$(1)="&" THEN GO TO 2550
2430 IF T$(1)="" THEN GO TO 2450
2432 LET U$=T$(1)
2435 IF (U$>="0" AND U$<="9") OR
     U$="." THEN LET C(K)=VAL (T$): GO TO
     2445
2440 LET C(K)=M(CODE (T$( TO 1))-64,VAL
     (T$(2 TO )))
2445 LET SP=SP+1: LET S(SP)=K
2450 NEXT K
2460 LET M(J,I)=C(K-1):RETURN
2500 LET C(K)=C(S(SP-1))+C(S(SP)): LET
     S(SP)=0: LET SP=SP-1: LET S(SP)=K:
     GO TO 2450
2510 LET C(K)=C(S(SP-1))-C(S(SP)): LET
     S(SP)=0; LET SP=SP-1: LET S(SP)=K:
     GO TO 2450
2520 LET C(K)=C(S(SP-1))*C(S(SP)): LET
     S(SP)=0: LET SP=SP-1: LET S(SP)= K:
     GO TO 2450
2530 LET C(K)=C(S(SP-1))/C(S(SP)): LET
     S(SP)=0: LET SP=SP-1: LET S(SP)=K:
     GO TO 2450
2540 LET C(K)=C(S(SP-1))^C(S(SP)): LET
     S(SP)=0: LET SP=SP-1: LET S(SP)=K:
     GO TO 2450
2550 LET C(K)=0-C(S(SP)): LET S(SP)=K: GO
     TO 2450
3010 >DIM H(4):DIM V(7):DIM S(20):DIM
     S$(20,5): DIM E$(20,5):DIM G$(20,5):
     DIM C(20)
```

Nueva savia para un viejo lenguaje

El FORTRAN ha experimentado numerosos cambios desde que fuera introducido a fines de los años cincuenta

Son dos las principales críticas que se han formulado respecto al FORTRAN como lenguaje de programación moderno: su falta de estructuras de control adecuadas, que hace difícil implementar programas de forma que resulten fáciles de comprender y corregir, y las facilidades muy restringidas para manipulación de caracteres. Con el correr de los años se han probado numerosas formas de superar estas deficiencias.

Un enfoque relativamente reciente consiste en utilizar un "preprocesador", una especie de compilador de nivel más alto. Los programas escritos en una versión ampliada de FORTRAN se ejecutan entonces con el preprocesador, que cambia todas las configuraciones adicionales a equivalentes en FORTRAN estándar. Esto ofrece la ventaja de poder escribir programas de una manera correctamente estructurada manteniendo, al mismo tiempo, una versión estándar y, en consecuencia, compatible. Los mejores ejemplos conocidos de preprocesadores de FORTRAN son el WATFOR y el RATFOR.

Cuando salió el FORTRAN 77, incorporó nuevas características en forma similar a los populares preprocesadores. Por este motivo el lenguaje conserva cierto grado de coherencia, si bien muchos compiladores para micros se basan en el FORTRAN IV con ampliaciones, en vez de en el auténtico FORTRAN 77.

El problema de proporcionar estructuras de control adecuadas es que el FORTRAN no posee una estructura de bloques como el ALGOL, el PASCAL o el C, y no existe ninguna forma de producir una sentencia compuesta encerrando entre paréntesis varias sentencias (con un begin...end, p.ej.). La única forma de aislar un bloque es extraerlo como una subrutina separada o bien rodearlo de GOTOs. Se ha introducido una nueva estructura de control, si bien no encaja muy bien en el resto del lenguaje. Se trata de IF...THEN...ELSE...ENDIF:

```
IF (expresión logica) THEN
          .........
          sentencias
          .........
ELSE
          .........
          sentencias
          .......................
ENDIF
```

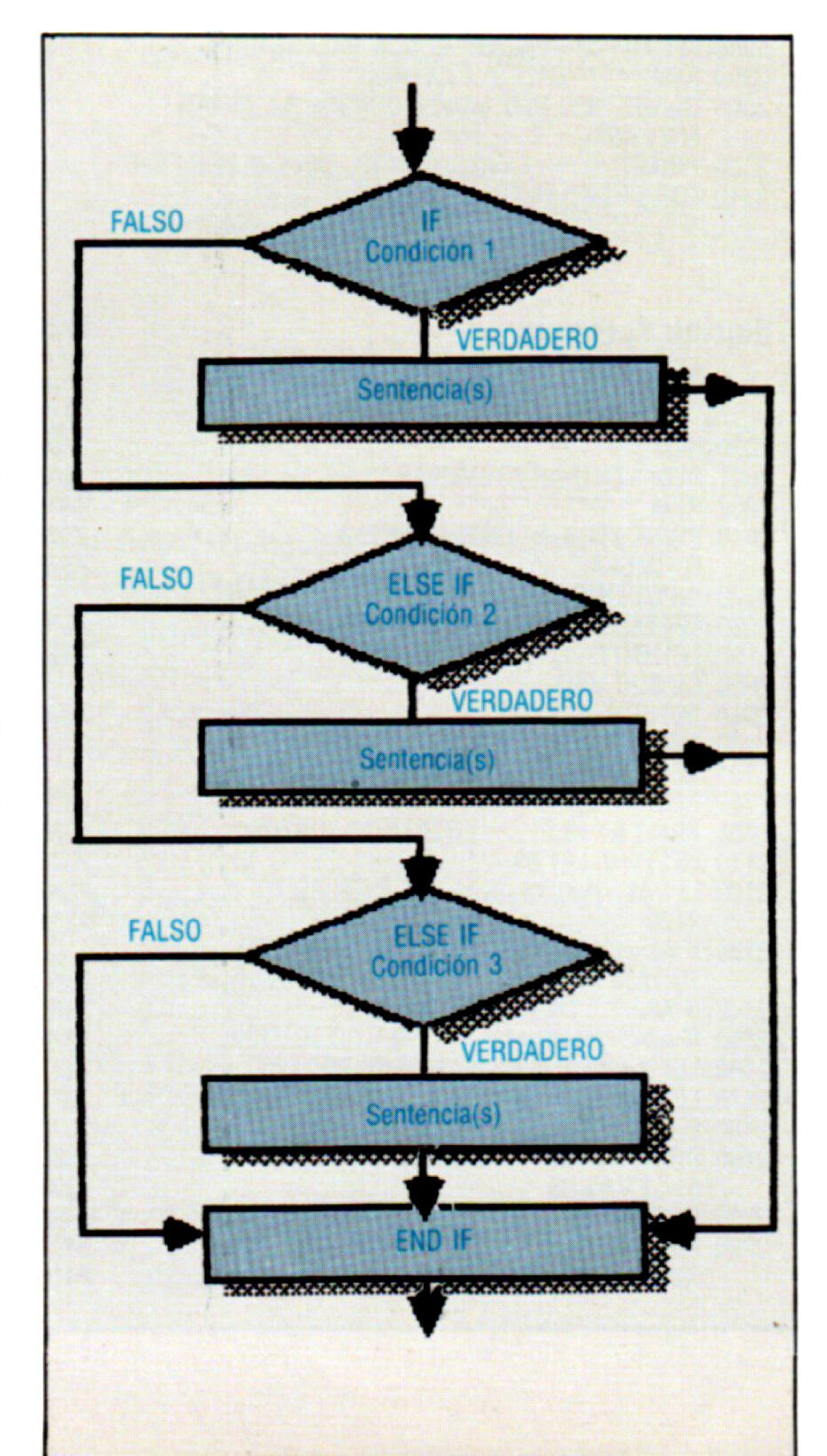
Las expresiones lógicas se forman exactamente de la misma forma que en FORTRAN estándar y, como es habitual, la parte ELSE se puede omitir.

Una forma muy útil de esta sentencia, que otros lenguajes harían bien en copiar, es una variante para abordar los IFs anidados:

```
IF (expresión logica) THEN
          .........
          sentencias
          .........
ELSEIF
          .........
          sentencias
          .........
ELSEIF
          .........
ELSE
          .........
ENDIF
```

en donde puede haber tantos ELSEIF como se requiera.

La segunda mejora importante ha sido la introducción de un tipo de datos de carácter. Las series de caracteres se declaran al principio del programa con las otras declaraciones en una de dos formas:



Nido de IFs
La mayoría de las versiones de BASIC admiten sentencias IF anidadas, pero el FORTRAN lo hace aún mejor al implementar la construcción ELSEIF, que permite anidar sentencias IF en cualquier profundidad a lo largo de varias líneas de programa. La construcción acaba con una sentencia ENDIF

```
CHARACTER*9                    CH1,CH2
CHARACTER                      CH3*7,CH4*16
```

Observe que se ha de dar una longitud máxima para la serie como *n, donde n es un entero. Ésta se puede aplicar ya sea a la palabra clave CHARACTER propiamente dicha, si todas las series tienen la misma longitud, o bien individualmente a cada serie nombrada. Estas declaraciones dan dos series de longitud 9, una de longitud 7 y una de longitud 16.

Se pueden declarar matrices de series del modo normal:

```
CHARACTER *4 CHARS(50)
```

que declara una matriz de 50 series de cuatro caracteres. A las variables en serie se les pueden asignar en serie, como en:

```
CH1'ABCDEFGHI'
```

Si la serie asignada es demasiado corta, se añadirán espacios en blanco; si es demasiado larga, se truncará lo que sobre.

Las series se pueden comparar utilizando los operadores de relación normales, como .GT., .LT., etc., pero también hay dos operadores nuevos para efectuar funciones de series. El operador subserie da acceso a cualquier secuencia de caracteres de la serie, como en:

```
CH1 (3:5)
```

Esto da la subserie de CH1 que comienza en el tercer carácter y termina en el quinto, o sea que:

```
CH1='ABCDEFGHI'
CH2=CH1 (3:5)
```

Mike Clowes

Descifrando el código

```
C     PROGRAMA PARA DECODIFICAR
C     MENSAJES SECRETOS OBSERVE
C     LA PRESENCIA DE UNA
C     SENTENCIA 'PROGRAM' EN FORTRAN 77
      INTEGER CONTADOR, PTRIN,
      PTROUT ARCHIVO LOGICO
      CHARACTER*10 NUM,CAR
      CHARACTER*30 IN,OUT
      DATA CONTADOR /0/
C
C     OBSERVE SENTENCIA 'DATA' PARA
C     INICIALIZAR UNA VARIABLE
C
C     SI ARCHIVO SECRETO NO EXISTE
C     IMPRIMIR MENSAJE ERROR
      INQUIERE (ARCHIVO
      'SECRETO',EXIST=ARCHIVO)
      IF(.NOT.ARCHIVO)THEN
        WRITE(1,10)
        STOP
      ENDIF
C
C     ABRIR ARCHIVOS
C
      OPEN(UNIT=12,ARCHIVO'MENSAJE',
        ESTADO='NUEVO'
C
C     ESPECIFICAR CLAVE DECODIFICADORA
C
      NUM='0123456789'
      CAR='MW3 0D$%"T'
C
C     LEER Y DESCIFRAR MENSAJE SECRETO
C
100   READ(11,20,END=1000)IN
        OUT=''
        PTROUT=1
        DO 200 I=1,30
          PTRIN=INDEX(CAR,IN(I:I))
C
C     LA FUNCION ESTANDAR 'INDEX'
C     DETERMINA LA POSICION
C     DE LA SUBSERIE IN(I:I) EN SERIE CAR
        IF(PTRIN.NE.O)THEN
          OUT(PTROUT:PTROUT)=NUM
          (PTRIN:PTRIN)
          PTROUT=PTROUT+1
        ENDIF
200   CONTINUE
      WRITE(12,20)OUT
C
C     OBSERVE EL USO DEL MISMO FORMATO
C     PARA ENTRADA Y SALIDA
C
      CUENTA=CUENTA+1
      GOTO 100
C
C     CERRAR
C     SE UTILIZA 'FINARCHIVO' PARA
C     ESCRIBIR FINAL DEL
C     MARCADOR DE ARCHIVO
1000  FINARCHIVO(UNIT=12)
      STOP
C
C     FORMATOS
10    FORMAT(1H,'EL ARCHIVO SECRETO NO
      EXISTE')
20    FORMAT(A)
C
C     OBSERVE QUE NO ES NECESARIO
C     ESPECIFICAR LA CANTIDAD
C     DE CARACTERES A ENTRAR
C
      END
```

Elemental, querido Watson
Este programa en FORTRAN lee datos desde un archivo denominado SECRETO y utiliza una clave de decodificación para descifrar texto codificado

Daría por resultado que CH2 contuviera 'CDE', o estrictamente 'CDE '. Observe que la segunda cifra dada es la posición del carácter final y no la longitud, lo que le resultaría más familiar al programador de BASIC.

El operador de concatenación, //, se utiliza para unir series entre sí, como en:

```
CHARACTER CH1*4,CH2*4,CH3*8
CH1='ABCD'
CH2='WXYZ'
CH3=CH1//CH2
```

que dejaría a CH3 conteniendo 'ABCDWXYZ'.

Archivos de datos

Un tema que aún nos queda por ver es el empleo de archivos de datos. Para sacar el máximo partido del FORTRAN se requiere un sistema con unidades de disco y la mayoría de los FORTRAN incluyen facilidades para el acceso a archivos tanto secuencial como directo. La verdadera fuente o destino de una operación de E/S se determina mediante el identificador de dispositivo en la sentencia READ o WRITE. Algunos números, por lo general los menores, se reservan para los dispositivos de E/S estándares tales como el teclado, la pantalla y la impresora; sin embargo, el programador dispondrá de una gama de números.

La sentencia OPEN se utiliza para asignarle un identificador a un archivo en disco.

Toma la forma:

```
OPEN(UNIT=expresion de enteros, FILE=
nombre del archivo, STATUS=estado)
```

El valor entero para la unidad (UNIT) es el número que se habrá de utilizar en subsiguientes sentencias READ o WRITE para acceder al archivo, y el nombre del archivo se atendrá a las convenciones del OS. El estado puede ser uno de varios valores; por ejemplo, OLD para un archivo de entrada que ya exista o NEW para un archivo de salida que esté en fase de creación.

En la sentencia OPEN hay varias cláusulas opcionales si se requiere algo que no sea un archivo secuencial directo. Éstas son :ACCESS=, para determinar acceso secuencial o directo; FORM=, para determinar si el archivo está formateado o sin formatear; IOSTAT=, que proporciona un medio de recuperación de errores si por algún motivo no se puede realizar la asignación de archivo correcta; y RECL=, para especificar una longitud de registro.

Una sentencia CLOSE similar podría ser:

```
CLOSE(UNIT=expresión de
enteros,STATUS=estado)
```

Ésta se puede utilizar para cerrar un archivo abierto, pero no siempre es necesario, puesto que cuando el programa termina los archivos se cierran automáticamente.

Además, existen otras muchas sentencias para manipulación de archivos que no hemos mencionado. Quizá la más útil de ellas sea INQUIRE, que le permite al programador determinar el estado actual y la actividad de cualquier archivo.

Hay, asimismo, numerosas adiciones útiles a las sentencias READ y WRITE, para usar cuando se efectúan operaciones de E/S con archivos.

Por ejemplo:

```
READ(7,10,REC=l)A,B,C
```

leería el primer registro de un archivo que se hubiera abierto para acceso directo.

Otro ejemplo:

```
READ(7,11,END=1000)A,B,C
```

haría que el control se transfiriera a la sentencia número 1000 cuando se llegara al final de un archivo secuencial.

Por último, veamos a los compiladores de FORTRAN propiamente dichos. Lo primero a destacar es que, por lo general, son rápidos y eficaces en operación y producen un código veloz, compacto y eficiente, lo que representa una ventaja fundamental, en especial para aplicaciones en tiempo real. Quizá esto sea lo que cabría esperar de un lenguaje que ha estado en servicio durante tanto tiempo y que es de un nivel lo bastante bajo para empezar con él.

La detección de errores no es, sin embargo, un punto fuerte de los compiladores de FORTRAN. Pueden aceptar bastante tranquilamente algunos errores garrafales que jamás pasarían, pongamos por caso, por un compilador de PASCAL. Los espacios, p. ej., no son significativos en una sentencia de FORTRAN, de modo que con frecuencia un compilador suprimirá todos los espacios de cada línea como primera tarea.

Un ejemplo de ello es la sentencia DO:

```
DO 100 I=1,100
```

Ésta establece un bucle a repetir 100 veces. No obstante, un punto en lugar de la coma, tras suprimir los espacios, lo deja a usted con:

```
DO100I=1.100
```

que es una asignación perfectamente legítima para una variable llamada DO100I. Se comenta que este simple error ha sido el origen de un desastre sumamente oneroso en el programa espacial norteamericano. El FORTRAN ha sido el lenguaje estándar para muchos trabajos de defensa, y errores como éste, al producirse en (entre otras cosas) sistemas de alerta nuclear, fueron factores fundamentales a la hora de decidir el desarrollo del ADA como lenguaje de recambio.

FORTRAN para micros
Los compiladores FORTRAN tienden a ser bastante caros y normalmente sólo los hay disponibles para máquinas de gestión; el paquete de la fotografía se ejecuta en el Tandy 2000. Además, el limitado espacio de memoria a menudo hace que la implementación del FORTRAN en micros sea difícil o incluso imposible.

No obstante, con la creciente disponibilidad de CP/M en ordenadores personales tales como Commodore 128, Amstrad CPC y otros, muchos usuarios encontrarán que existe una versión de FORTRAN para su máquina

MS-FORTRAN cortesía de Computer World, St Giles High Street, London WC2



Queridos residentes

Veamos cómo, gracias a las extensiones residentes de sistema, se pueden añadir nuevas instrucciones al BASIC Locomotive

En el corazón del OS del Amstrad encontramos el núcleo. Esta sección del firmware es la que se encarga del mantenimiento interno del ordenador, como el procesamiento de eventos e interrupciones y la gestión de la memoria.

Muchas entradas al núcleo se realizan a través de las direcciones que están entre &BCC8 y &BD10, pero el núcleo está provisto de su propio bloque, dividido en una sección superior y otra inferior, que se hallan en &B900-&B921 y &0000-&003B, respectivamente. Pero a diferencia del área del bloque de saltos del sistema central, estas posiciones no podrán ser parcheadas por el usuario. Algunas de las entradas son de un uso frecuente, y en la tabla *Direcciones útiles del núcleo* se da una breve descripción de las más interesantes.

Una característica muy poderosa del núcleo es su capacidad para procesar instrucciones externas. Estas instrucciones están en las ROM o en la RAM. En caso de que estén en la RAM, estas instrucciones son conocidas como *extensiones residentes del sistema* (RSX). Una instrucción externa puede ser una rutina de cualquier tamaño o función. Por ejemplo, instrucciones como |DIR y |ERA y otras que incluye el AMSDOS son rutinas que tratan los archivos del disco. Adicionalmente, todo el lenguaje del BASIC es considerado por el sistema como una RSX (pruebe a teclear |BASIC para ver qué pasa).

Las instrucciones externas pueden establecerse de dos maneras. Primera, en el momento de la puesta en marcha, en que se cargan las instrucciones desde la ROM. Segunda, bajo el control de programa, desde donde se pueden cargar estas instrucciones ya sea desde la ROM, ya sea desde la RAM. En ambos casos una instrucción externa está reseñada en una tabla de instrucciones, cuyo formato se muestra en el diagrama. Los nombres de instrucciones pueden tener hasta 16 caracteres, con el bit 7 en uno. Los caracteres pueden ser cualquier código de siete bits, pero si la instrucción ha de llamarse desde el BASIC habremos de cuidar de que los caracteres empleados sean accesibles desde el teclado.

Si las instrucciones están en forma de una RSX (es decir, cargadas en la RAM) entonces el proceso efectivo de *registro* (aviso al firmware de su presencia) se realiza mediante la rutina de núcleo KL_LOG_EXT. Esta rutina es llamada a través de &BCD1, con la dirección de la tabla de instrucciones en BC. Por su parte, HL debe contener la dirección de un espacio de trabajo de cuatro bytes que el núcleo pueda usar, y ambas direcciones deben encontrarse en los 32 K centrales de la RAM.

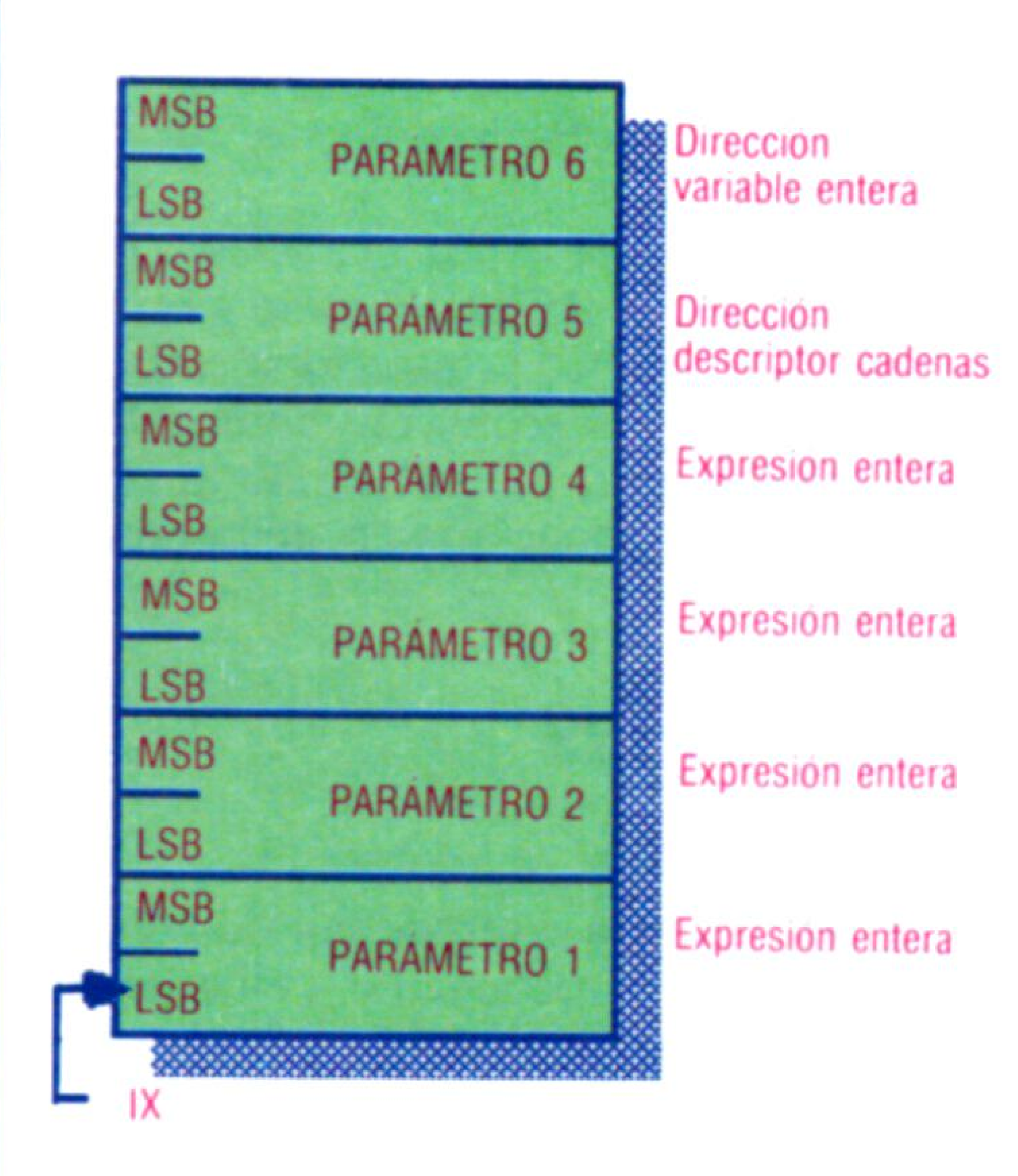
En los casos en que las instrucciones son almacenadas en la ROM, lo que se presenta al firmware es la ROM misma y no la tabla de instrucciones. La ROM debe también adecuarse a un formato particular, como se muestra en la figura tercera. Las ROM pueden ser designadas como ROM de primer plano, de fondo o de extensión. Se emplean ROM de primer plano para albergar programas que, cuando son entrados, toman el control del firmware. Por ejemplo, la ROM del BASIC es de este tipo, como lo sería la ROM de cualquier otro lenguaje, como el PASCAL.

Una ROM de fondo contiene las instrucciones externas que pueden estar o no registradas por el programa de primer plano. Finalmente, una ROM de extensión sirve para albergar el código para un programa de primer plano o una rutina de instrucción externa que pueda no ajustarse a la ROM existente.

Cada ROM se direcciona en el intervalo 0-251, aunque cada tipo tiene sus condiciones restrictivas particulares. Las ROM de primer plano pueden ocupar una dirección cualquiera, siempre que todas las direcciones que quedan por debajo de la escogida sean ocupadas por otras ROM del tipo que sea, dado que el núcleo busca las ROM ascendiendo desde la dirección 0 hasta que encuentra el primer espacio vacío. Las ROM de fondo deben estar en las direcciones 1-7 en el 464, y 1-15 en el 664 y 6128. Las ROM de extensión pueden ocupar cualquier dirección.

Por ejemplo, si el sistema tiene ya el FORTH en la ROM 0 (por defecto) y la ROM de fondo en la ROM 1, entonces una segunda ROM de primer plano deberá instalarse en la dirección 2 de ROM

Tabla de parámetros



Proceso de entrada
A la entrada en una rutina de instrucción llamada desde el BASIC, todo parámetro pasado por el usuario es almacenado en un bloque de parámetros. La dirección de base del bloque queda apuntada por IX y un registro A contiene el número de parámetros que se ha pasado. Esta figura muestra la estructura del bloque establecida después de ejecutar | COMMAND, 5,24.6, a%,x,@b$,@x%

Caroline Clayton

Instrucciones externas



Establecimiento de la tabla
Las instrucciones externas son referenciadas a través de una tabla de instrucciones. Una tabla puede contener datos para varias instrucciones diferentes agrupadas en un todo y el formato de dicha tabla es el que mostramos aquí

para asegurar la continuidad. Sin embargo, podría introducirse una segunda ROM de fondo en cualquier dirección entre 3 y 7 (15 en los 664/6128) e instalarse una extensión a cualquier dirección.

Todas las ROM de fondo son registradas al poner en funcionamiento la máquina. Las ROM de fondo pueden ser inicializadas bien individualmente mediante KL_INIT_BACK, bien conjuntamente mediante KL_ROM_WALK. La ROM del BASIC inicializa, por otro lado, todas las ROM de fondo cuando es introducida.

Cualquier ROM de extensión ocupa la misma área de memoria que la ROM del BASIC (de &C000 a &FFFF).

Antes de que se pueda entrar una ROM, primero se debe desactivar la ROM del BASIC y activar la ROM adecuada. El núcleo proporciona varias rutinas para conectar y desconectar ambas ROM, la superior y la inferior, lo cual exime al programador de pensar en conectarlas o desconectarlas directamente.

Posición de las instrucciones

Cuando las instrucciones externas han sido registradas se puede acceder directamente a ellas desde el BASIC, como veremos más adelante, o bien a través del núcleo mediante KL_FIND_COMMAND. Esta entrada toma el nombre de una instrucción y busca las RSX y ROM de fondo para su acoplamiento. Si se encuentra la instrucción, entonces la dirección de la entrada de la rutina se da en HL, conteniendo el registro C a su vez el número de la ROM de fondo (que puede ser ignorado si la rutina de instrucción está en la RAM). Si no se encuentra ninguna instrucción, la rutina retorna con el arrastre falso. Esta rutina es llamada en &BCD4, con HL conteniendo la dirección del nombre de la instrucción, que ha de acabar con un carácter cuyo bit 7 esté a uno.

Las instrucciones externas pueden ser entradas simplemente desde el BASIC antecediendo el nombre de la instrucción con el símbolo | (@ con *shift*). El BASIC permite también pasar parámetros a la rutina separándolos con comas, por Ejemplo:

```
|INSTRUCCION, 1,2,3,x,y*3,@$,@X%
```

El firmware pasa entonces el control a la rutina de instrucción, con el número de parámetros retenidos en el registro A e IX apuntando a la tabla que contiene la información de dichos parámetros. Esta tabla contiene dos bytes por cada parámetro, almacenados en el formato siguiente:

Números enteros/ expresiones	– enteros con signo de 16 bits
Números reales	– enteros sin signo puestos a 16 bits

Las direcciones de variables numéricas o de cadenas pueden ser pasadas también empleando el símbolo @, dejando las variables enteras una dirección de dos bytes donde se almacenan, y las variables de cadena la dirección del bloque descriptor de cadenas. Por desgracia, no es posible determinar si la entrada en la tabla es la dirección de un descriptor de cadenas o de un entero absoluto: es cosa de la rutina de instrucción decidir el tipo de parámetro que espera.

A la entrada a la rutina de instrucción, IX apunta al LSB (bit menos significativo) del *último* parámetro especificado (en el ejemplo anterior, sería la dirección de X%). El nuevo incremento de IX indexará, por tanto, el LSB del penúltimo parámetro pasado. El diagrama (p. 2137) ilustra cómo se indexa la tabla en el ejemplo dado.

A menudo es preferible mirar al comienzo el primer parámetro especificado y trabajar a través de ellos según van entrando, antes que hacerlo en orden inverso. El primer listado proporciona una breve utilidad que modifica IX para que apunte al

Direc. útiles del núcleo

&BCC8 RESET	Restaura el núcleo: limpia evento, colas, etc.
&BCCB ROM WALK	Inicializa todas las ROM de fondo
&BCD1 LOG EXT	Registra una tabla de instrucciones RSX
&BCD4 FIND COMMAND	Posiciona la dir. de RSX, ROM fondo y primer plano
&B90F ROM SELECT	Selecciona una determinada ROM superior
&B912 CURR SELECTION	Comprueba qué ROM sup. es la seleccionada



LSB del primer parámetro, en vez del último. Obsérvese que en este caso es necesario, para mirar al MSB (bit más significativo) del primer parámetro, incrementar una vez el IX, mientras que para mirar al MSB del segundo parámetro será decrementado una vez. De nuevo habrá de decrementarse para mirar al LSB del segundo parámetro, y así sucesivamente.

Las instrucciones RSX son relativamente sencillas para que las emplee un programador en lenguaje máquina, y presentan la gran ventaja de que son fáciles para el usuario comparadas con la necesidad, en otras máquinas, de emplear instrucciones como CALL o SYS. Por esta razón, la instrucción CALL en los ordenadores Amstrad CPC adquiere menor relevancia, y el esfuerzo suplementario que se requiere para implementar las extensiones residentes de sistema compensa con creces. Como ejemplo, damos aquí un listado para una RSX que permite llamar las rutinas del firmware mencionadas en estos capítulos directamente desde el BASIC, con los registros establecidos según se requieren.

ROM de ampliación

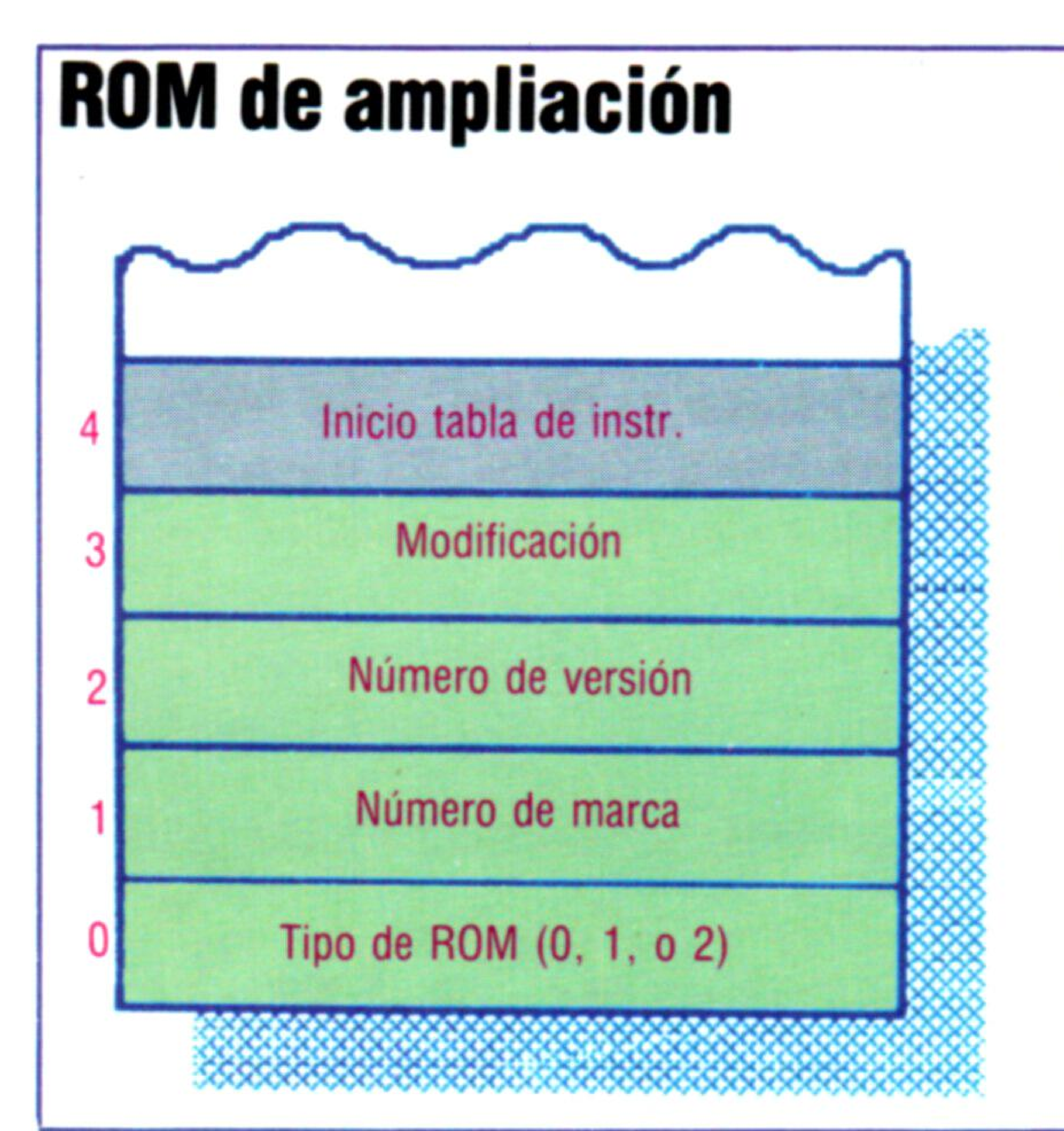


Tabla de la ROM
La tabla muestra un esquema de los bytes empleados por el sistema operativo para reconocer y "registrar" las ROM individuales de ampliación. Una interrogación posterior del sistema operativo posibilitará el retorno de las direcciones para las rutinas de instrucción definidas en la tabla de instrucciones externas de la ROM

Pasar el control

Puntero de parámetros

Esta breve rutina altera IX al entrar a una rutina de instrucción externa para que apunte al primer parámetro especificado

```
          ;entry:   A contiene el número de parámetros
          ;         IX contiene la dirección de la tabla
          ;exit:    IX contiene              la dirección
          ;         de la 1.ª                entrada en tabla
          ;         DE se altera,            los restantes regs.
          ;         se conservan
          ;
5F                  ld    e,a                ;copia núm. para E
1D                  dec   e                  ;num. de pars.-1
CB23                sla   e                  ; *2
1600                ld    d,0                ;toma desp.en DE
DD19                add   ix,de              ;lo suma para inic.
          ; IX apunta ahora al LSB de la 1.ª entrada
          ; en la tabla de parámetros
```

Llamada del firmware

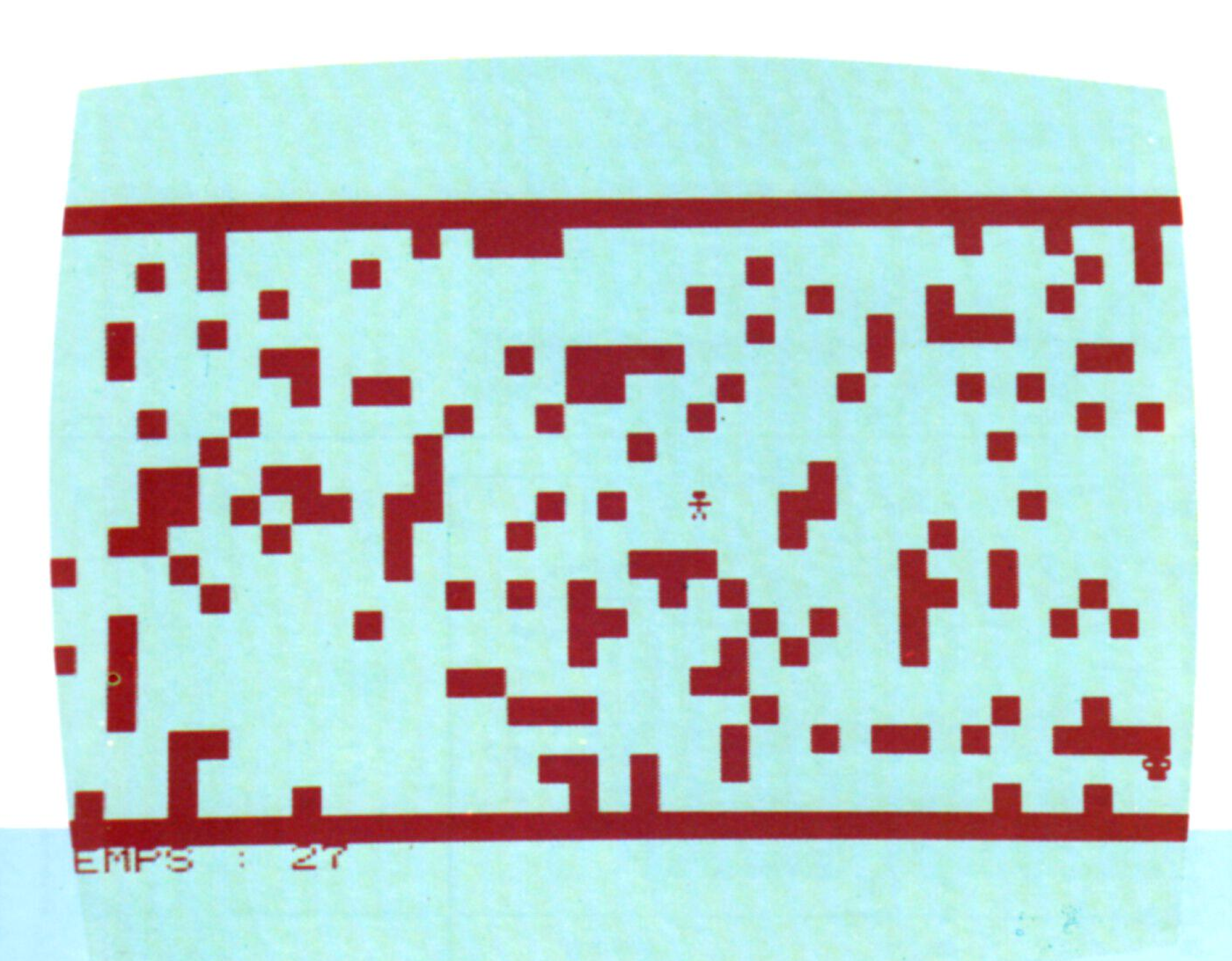
Este listado proporciona una RSX que permite la llamada del firmware directamente desde el BASIC, con los registros establecidos a voluntad. La sintaxis de la instrucción es la siguiente:

```
   |FIRMCALL,<direccion entrada>,
[,<A> <HL>[,<BC>[,DE]]]]
```

```
          k1__log:    equ   #bcd1
01902C    logon:      ld    bc,commands      ;apunta a tabla
219E2C                ld    hl,scratch       ;y cuad. apuntes
CDD1BC                call  kl__log          ;registra instr.
C9                    ret                    ;y eso es todo
952C      comman:     defw  name             ;apunta al nombre
C3A22C                jp    firmcall         ;punto de entrada
4649524D  name:       defm  "FIRMCAL"        ;pone a 1 el bit 7
                                             del...
CC                    defb  "L"+#80          ;..ult. byte del n.
00                    defb  0
          scratc:     defs  4                ;res. area datos
          ; IFIRMCALL,      entry, a, hl, bc, de
5F        firmca:     ld    e,a              ;copia numero
                                             para E
1D                    dec   e                ;num. paramtr-1
CB23                  sla   e                ;*2
1600                  ld    d,0              ;toma desp. en DE
DD19                  add   ix,de            ;lo añade a inicio
DDE5                  push  ix
E1                    pop   hl
FE06                  cp    6                ;hasta 5 pars.
D0                    ret   nc               ;muchos, no seguir
B7                    or    a                ;hay parametros?
C8                    ret   z                ;no, pues no
                                             seguir
47                    ld    b,a              ;guardar cont. en B
23                    inc   hl               ;toma MSB de dir.
7E                    ld    a,(hl)           ;lo lee
32E62C                ld    (entry+1),a      ;lo guarda
2B                    dec   hl               ;LSB
7E                    ld    a,(hl)
32E52C                ld    (entry),a
2B                    dec   hl               ;MSB de AF
2B                    dec   hl               ;LSB de AF
05                    dec   b                ;bastante?
2822                  jr    z,call           ;si, realizar llam.
7E                    ld    a,(hl)           ;si no, leer en A
05                    dec   b                ;bastante?
281E                  jr    z,call           ;si, realizar llam.
2B                    dec   hl               ;MSB de HL
56                    ld    d,(hl)
2B                    dec   hl
5E                    ld    e,(hl)
EB                    ex    de,hl            ;toma en HL
05                    dec   b                ;bastante?
2816                  jr    z,call           ;si, realizar llam.
EB                    ex    de,hl            ;retomar punt.
2B                    dec   hl               ;MSB de BC
F5                    push  af               ;guarda A de mom.
7E                    ld    a,(hl)
2B                    dec   hl               ;LSB de BC
4E                    ld    c,(hl)
05                    dec   b                ;bastante
47                    ld    b,a              ;establece BC en
                                             todo caso
EB                    ex    de,hl            ;y HL
2003                  jr    nz,miss          ;toma el ultimo
F1                    pop   af               ;restaura la pila
1808                  jr    call             ;y hace la llamada
F1        miss:       pop   af
EB                    ex    de,hl            ;retoma la tabla
D5                    push  de               ;guarda valor HL
2B                    dec   hl               ;MSB de DE
56                    ld    d,(hl)
2B                    dec   hl               ;LSB de DE
5E                    ld    e,(hl)
E1                    pop   hl               ;retoma HL
C3        call:       defb  #c3              ;instr. de salto
0000      entry:      defw  0                ;direccion de salto
          ;se ha parcheado la dirección del salto
          ;a la rutina
```

Persecución

El ladrón, representado por un as de piques, ha escapado llevándose el botín... Se trata de un tema muy visitado por la informática lúdica. El listado es para los micros MSX



El amigo de lo ajeno se esconde en la ciudad, y usted tiene 50 minutos para encontrarlo y detenerlo. Atención, ¡no se precipite! En efecto, si se echa sobre él sin pensar, el fugitivo tiene todas las posibilidades de escapársele. La mejor manera de atraparlo es alcanzándolo de lado. (Es el método más efectivo a condición de no fallar.) Si no se siente lo bastante seguro de sí mismo, atáquelo de frente, lo cual es más fácil pero mucho menos eficaz, ya que no es tan discreto. Otro consejo: no intente perseguirlo; no daría resultado, pues él es mucho más rápido que usted. Ha de observar sus movimientos como si fuera un detective. Cuando vea que da la vuelta, acérquese sigilosamente y sorpréndalo en el momento justo. Pero recuerde: ¡el tiempo va pasando!

```
10 REM ***************
20 REM * PERSECUCION *
30 REM ***************
40 KEY OFF
50 WIDTH 39
60 GOSUB 1060
70 S=0
80 N=32
90 V=128
100 P=129
110 GOSUB 670
120 ST=STICK (0)
130 DX=(ST=7)-(ST=3)
140 DY=(ST=1)-(ST=5)
150 Z=Z-.2
160 LOCATE 0,24,0
170 PRINT "TIEMPO :";INT (Z+1);
180 IF Z<0 THEN 370
190 PX=PX+DX
200 PY=PY+DY
210 C=VPEEK (PX+PY*40)
220 IF C=V THEN 1010
230 IF C<>32 THEN PX=XP:PY=YP
240 VPOKE XP+YP*40,N
250 VPOKE PX+PY*40,P
260 YP=PY
270 XP=PX
280 VX=VX+CX
290 VY=VY+CY
300 IF VPEEK (VX+VY*40)<>N THEN GOSUB 510
310 IF VPEEK (VX+VY*40)<>N THEN 280
320 VPOKE XV+YV*40,N
330 VPOKE VX+VY*40,V
340 XV=VX
350 YV=VY
360 GOTO 120
370 IF INKEY$<>"" THEN 370
380 FOR I=1 TO 1000
390 NEXT I
400 LOCATE 10,6
410 PRINT "TIEMPO TRANSCURRIDO";
420 LOCATE 10,10
430 PRINT "PUNTOS :";S;
440 LOCATE 10,18,1
450 PRINT "OTRA ?";
460 D$=INKEY$
470 IF D$="" THEN 460
480 IF D$<>"N" AND D$<>"n" THEN RUN
490 CLS
500 END
510 DT=DT+1
520 GOSUB 620
530 IF VPEEK(XV+CX+(YV+CY)*40)=N THEN VX
    =XV+CX:VY=YV+CY:RETURN
540 DT=DT-2
550 GOSUB 620
560 IF VPEEK (XV+CX+(YV+CY)*40)=N THEN VX
    =XV+CX:VY=YV+CY:RETURN
570 DT=DT-1
580 GOSUB 620
590 VX=XV+CX
600 VY=YV+CY
610 RETURN
620 IF DT> 4 THEN DT=DT-4
630 IF DT< 1 HEN DT=DT+4
640 CX=(DT=1)-(DT=3)
650 CY=(DT=2)-(DT=4)
660 RETURN
670 CLS
680 FOR VX=0 TO 39
690 VPOKE VX+40,219
700 VPOKE VX+880,219
710 NEXT VX
720 FOR VY=2 TO 22
730 VPOKE VY*40,219
740 VPOKE VY*40+39,219
750 NEXT VY
760 FOR VX=1 TO 150
770 GOSUB 970
780 VPOKE PX+PY*40,219
790 NEXT VX
800 GOSUB 970
810 VX=PX
820 VY=PY
830 VPOKE VX+VY*40,V
840 XV=VX
850 YV=VY
860 GOSUB 970
870 VPOKE PX+PY*40,P
880 XP=PX
890 YP=PY
900 Z=50
910 CX=0
920 CY=0
930 DX=0
940 DY=0
950 DT=0
960 RETURN
970 PX=INT (RND(1)*37)+1
980 PY=INT (RND(1)*21)+2
990 IF VPEEK (PX+PY*40)<>N THEN 970
1000 RETURN
1010 FOR I=1 TO 5
1020 PLAY "DC"
1030 NEXT I
1040 S=S+1
1050 GOTO 110
1060 SCREEN 0,0
1070 COLOR 6,14
1080 CLS
1090 DEFINT A-Y
1100 PX=RND(-TIME)
1110 RESTORE
1120 FOR I=0 TO 15
1130 READ A
1140 VPOKE 3072+I,A*4
1150 NEXT I
1160 RETURN
1170 DATA 30,63,45,63,30,30,30,0
1180 DATA 28,28,8,62,8,8,20,20
```



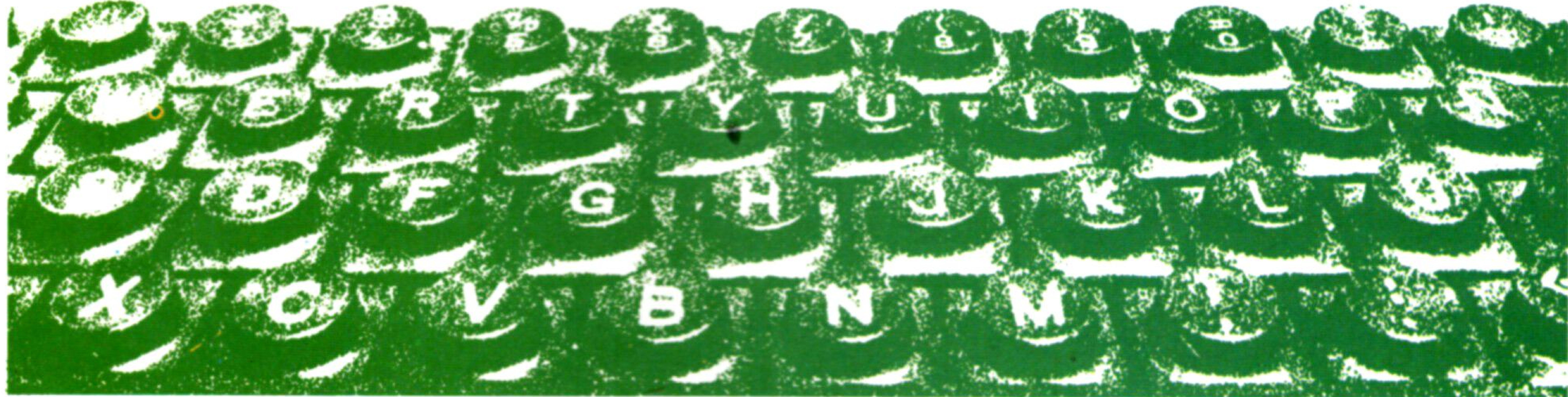
Con el próximo fascículo se pondrán a la venta las tapas correspondientes al noveno volumen.



El juego de tapas va acompañado de un sobre con los transferibles, numerados del 1 al 8, correspondientes a los volúmenes de que consta la obra; esto le permitirá marcar el lomo de cada uno de los volúmenes, a medida que aumente su colección.

Para encuadernar los 12 fascículos que componen un volumen, es preciso arrancar previamente las cubiertas de los mismos.

No olvide que, antes de colocar los fascículos en las tapas intercambiables, debe usted estampar el número en el lomo de las mismas, siguiendo las instrucciones que se dan a continuación:



1 Desprenda la hojita de protección y aplique el transferible en el lomo de la cubierta, haciendo coincidir los ángulos de referencia con los del recuadro del lomo.

2 Con un bolígrafo o un objeto de punta roma, repase varias veces el número, presionando como si quisiera borrarlo por completo.

3 Retire con cuidado y comprobará que el número ya está impreso en la cubierta. Cúbralo con la hojita de protección y repita la operación anterior con un objeto liso y redondeado, a fin de asegurar una perfecta y total adherencia.



Cada sobre de transferibles contiene una serie completa de números, del 1 al 10, para fijar a los lomos de los volúmenes. Ya que en cada volumen sólo aplicará el número correspondiente, puede utilizar los restantes para hacer una prueba preliminar.

Con el próximo número se ponen a la venta las tapas intercambiables para encuadernar 12 fascículos de

mi COMPUTER

Cada juego de tapas va acompañado de una colección de transferibles, para que usted mismo pueda colocar en cada lomo el número de tomo que corresponda



10107
9 788485 822836